



Universidad
Carlos III de Madrid

DEPARTAMENTO DE INFORMÁTICA

PROYECTO FIN DE CARRERA



MODELADO DE PATRONES MELÓDICOS MEDIANTE TRIES

Autor: Alberto Gómez Bravo

Tutores: Tomás Eduardo de la Rosa Turbides
Sergio Jiménez Celorrio

Leganés, 12 de Diciembre de 2011

Título: Modelado de Patrones Melódicos mediante Tries.

Autor: Alberto Gómez Bravo.

Director: Tomás Eduardo de la Rosa Turbides.

Sergio Jiménez Celorrio.

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 12 de Diciembre de 2011 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO

PRESIDENTE

Agradecimientos

En primer lugar, me gustaría agradecer a mis padres, a mi hermana Virginia y a Jaime su apoyo durante todos estos años en la Universidad.

También me gustaría agradecer a mis amigos de siempre los buenos momentos pasados, al igual que a mis compañeros de estudios, sin cuyos conocimientos y amistad estos años habrían sido muy diferentes.

Por último, agradecer a mis tutores su paciencia y trabajo durante el desarrollo del proyecto.

Resumen

El presente proyecto trata de extraer patrones reconocibles de un estilo musical con el fin de crear melodías pertenecientes a dicho estilo.

El sistema creado permite generar patrones de una duración arbitraria y un estilo musical concreto. A partir de su concatenación se crean melodías MIDI correspondientes al estilo modelado.

El desarrollo del proyecto se ha llevado a cabo utilizando el lenguaje de programación JAVA, el paquete Javax.Sound para el manejo de archivos de audio y la clase Trie del paquete key.util para el manejo de la estructura de datos de idéntico nombre.

Palabras clave: Inteligencia Artificial, Música, Modelado automático, Composición automática, Extracción de patrones, Tries.

Índice general

1 Introducción y objetivos	11
1.1 Introducción	11
1.2 Objetivos	12
1.3 Fases del desarrollo	13
1.4 Estructura de la memoria	15
2 Medios empleados	16
2.1 Teoría Musical	16
2.2 MIDI	20
2.3 MusicXML	23
2.4 Java	26
2.5 Tries	31
3 Estado del Arte	34
3.1 Sistemas de composición musical	34
4 Planificación y presupuesto	38
4.1 Planificación y metodología de desarrollo	38
4.2 Presupuesto	41
5 Trabajo realizado	43
5.1 Flujo del sistema	43
5.2 Arquitectura del sistema	50
5.2.1 Paquete Principal	51
5.2.2 Paquete Extraer	52
5.2.3 Paquete Trie	57
5.2.4 Paquete Interfaz	59
6 Resultados experimentales	63
6.1 Música clásica	63
Experimento 0: Melodía formada por un solo patrón.	65
Experimento 1: Melodía formada por varios patrones	67
Experimento 2: Melodía formada por varios patrones de duración corta	71
Experimento 3: Melodía formada con obras de distintos autores por patrones de duración corta	74
Conclusiones Música Clásica	78
6.2 Metal	79

Experimento 4: Melodía formada por patrones de duración corta obtenidos de una sola obra.	80
Experimento 5: Melodía formada por patrones de duración corta obtenidos de varias obras.	83
Conclusiones Metal.....	85
6.3 Blues.....	86
Experimento 6: Melodía formada por patrones de duración corta obtenidos de una sola obra.	87
Experimento 7: Melodía formada por patrones de duración corta obtenidos de varias obras.	90
Conclusiones Blues	92
7 Conclusiones	93
7.1 Líneas futuras	95
Referencias.....	97
Apéndice 1: Manual de Usuario.....	99
Apéndice 2: Experimentos no incluidos en los resultados.....	105
Experimento 8: Melodía formada por un solo patrón con obras de distintos autores	105
Experimento 9: Melodía formada por patrones de duración media con obras de distintos autores.	107

Índice de Figuras

Figura 1 Organización vertical y horizontal [Ltm]	16
Figura 2 Altura [Wik].....	17
Figura 3 Duración	17
Figura 4 Acorde fundamental [Sax].....	19
Figura 5 Intervalo musical [Wik].....	19
Figura 6 Flujo de datos MIDI [Css]	21
Figura 7 Ejemplo MusicXML [Wik]	25
Figura 8 Entradas y salidas del paquete Player	27
Figura 9 Modelo Vista-Controlador [Dav]	29
Figura 10 Ejemplo JFileChooser.....	30
Figura 11 Ejemplo JTextField.....	30
Figura 12 Ejemplo JButton	31
Figura 13 Ejemplo JList	31
Figura 14 Ejemplo de trie.....	32
Figura 15 Diagrama Gantt de los módulos desarrollados	39
Figura 16 Ejemplo de Intervalo de duración [Mus]	43
Figura 17 Ejemplo de frase de intervalos de duración.....	43
Figura 18 Ejemplo de Intervalo de Altura [Mus].....	44
Figura 19 Ejemplo de frase de intervalos de altura.....	44
Figura 20 Salida de un trie de intervalos de altura obtenido de la obra Preludio 6.....	45
Figura 21 Salida de un trie de intervalos de duración obtenido de la obra Only for the weak	46
Figura 22 Algoritmo de generación de frases	47
Figura 23 Algoritmo de generación de un patrón	48
Figura 24 Relación de paquetes que componen el sistema	50
Figura 25 Clase Main	51
Figura 26 Diagrama de clases módulo Extraer	52
Figura 27 Clase Nota.....	53
Figura 28 Clase Frase.....	53
Figura 29 Clase Cancion	54
Figura 30 Clase IntervaloDuracion	54
Figura 31 Clase IntervaloAltura.....	55

Figura 32 Clase Lectura	56
Figura 33 Clase Melodías.....	56
Figura 34 Clase Patrones.....	57
Figura 35 Clase Directorio	57
Figura 36 Clase Trie.....	57
Figura 37 Paquete Interfaz	59
Figura 38 Selección de estilo	60
Figura 39 Elección de directorio	60
Figura 40 Creación de patrones y estructura de la melodía	61
Figura 41 Generar la melodía.....	62
Figura 42 Partitura original de La Marcha Turca.....	64
Figura 43 Melodía generada a partir de La Marcha Turca.....	65
Figura 44 Fragmento de la melodía generada a partir de La Marcha Turca	66
Figura 45 Fragmento de La Marcha Turca	66
Figura 46 Fragmento segundo de la melodía generada a partir de La Marcha Turca.....	66
Figura 47 Fragmento segundo de La Marcha Turca	67
Figura 48 Patrón A Experimento 1	68
Figura 49 Patrón B Experimento 1.....	68
Figura 50 Patrón C Experimento 1.....	68
Figura 51 Patrón D Experimento 1	69
Figura 52 Melodía obtenida Experimento 1.....	70
Figura 53 Patrón A Experimento 2	72
Figura 54 Patrón B Experimento 2.....	72
Figura 55 Patrón C Experimento 2.....	72
Figura 56 Patrón D Experimento 2	72
Figura 57 Melodía obtenida Experimento 2.....	73
Figura 58 Patrón A Experimento 3	75
Figura 59 Patrón B Experimento 3.....	75
Figura 60 Patrón C Experimento 3.....	75
Figura 61 Patrón D Experimento 3	76
Figura 62 Melodía obtenida Experimento 3.....	76
Figura 63 Patrón A Experimento 4	80
Figura 64 Patrón B Experimento 4.....	80
Figura 65 Patrón C Experimento 4.....	80
Figura 66 Patrón D Experimento 4	80
Figura 67 Melodía obtenida Experimento 4.....	81
Figura 68 Fragmento de la partitura de Davidian	82
Figura 69 Fragmento de la Melodía obtenida en Experimento 4.....	82
Figura 70 Fragmento de la partitura de Davidian	82
Figura 71 Fragmento de la Melodía obtenida en Experimento 4.....	82
Figura 72 Patrón A Experimento 5	83
Figura 73 Patrón B Experimento 5.....	83
Figura 74 Patrón C Experimento 5.....	83
Figura 75 Patrón D Experimento 5	84
Figura 76 Melodía obtenida Experimento 5.....	84
Figura 77 Patrón a Experimento 6	87
Figura 78 Patrón B Experimento 6.....	87
Figura 79 Patrón C Experimento 6.....	87

Figura 80 Patrón D Experimento 6	87
Figura 81 Melodía obtenida en Experimento 6.....	88
Figura 82 Fragmento de la partitura de Take Five.....	89
Figura 83 Fragmento de la Melodía generada en Experimento 6	89
Figura 84 Fragmento de la Melodía generada en Experimento 6	89
Figura 85 Fragmento de la partitura de Take Five	89
Figura 86 Patrón A Experimento 7	90
Figura 87 Patrón C Experimento 7.....	90
Figura 88 Patrón C Experimento 7.....	90
Figura 89 Patrón D Experimento 7	91
Figura 90 Melodía obtenida en Experimento 7	91
Figura 91 Partituras MusicXml utilizadas.....	99
Figura 92 Pasos para ejecutar.....	100
Figura 93 Interfaz de usuario	101
Figura 94 Elección de estilo	101
Figura 95 Interfaz con patrones ya creados.....	102
Figura 96 Interfaz con secuencia de patrones elegidos.....	103
Figura 97 Interfaz con melodía generada.....	103
Figura 98 Resultado final	104
Figura 99 Melodía obtenida en Experimento 8.....	105
Figura 100 Fragmento primero Experimento 8.....	106
Figura 101 Fragmento segundo Experimento 8.....	106
Figura 102 Fragmento tercero Experimento 8	106
Figura 103 Patrón A Experimento 9	107
Figura 104 Patrón B Experimento 9.....	107
Figura 105 Patrón C Experimento 9.....	108
Figura 106 Patrón D Experimento 9	108
Figura 107 Melodía obtenida Experimento 9.....	110

Índice de Tablas

Tabla 1 Tipos de intervalos musicales	20
Tabla 2 Tipos de mensajes MIDI	22
Tabla 3 Tabla de valores COCOMO	40
Tabla 4 Gastos contratación de personal	42
Tabla 5 Gastos material fungible	42
Tabla 6 Datos extraídos del trie de intervalos de altura de La Marcha Turca	63
Tabla 7 Datos extraídos del trie de intervalos de duración de La Marcha Turca.....	63
Tabla 8 Datos extraídos del trie de intervalos de altura con obras de Música Clásica	75
Tabla 9 Datos extraídos del trie de intervalos de duración con obras de Música Clásica	75
Tabla 10 Datos extraídos del trie de intervalos de altura de la obra Davidian.....	79
Tabla 11 Datos extraídos del trie de intervalos de duración de la obra Davidian.....	79
Tabla 12 Datos extraídos del trie de intervalos de altura con obras de Metal	79
Tabla 13 Datos extraídos del trie de intervalos de duración con obras de Metal.....	79
Tabla 14 Datos extraídos del trie de intervalos de altura de la obra Take Five	86
Tabla 15 Datos extraídos del trie de intervalos de duración de la obra Take Five	86
Tabla 16 Datos extraídos del trie de intervalos de altura de obras de Blues.....	86
Tabla 17 Datos extraídos del trie de intervalos de duración con obras de Blues.....	86

1 Introducción y objetivos

En este apartado se va a dar una primera introducción al proyecto, describiendo sus objetivos y las fases de desarrollo por las que se ha pasado.

1.1 Introducción

El presente proyecto queda incluido en el marco de la Inteligencia Artificial, más concretamente trataremos el modelado automático. Este modelado se llevará a cabo mediante una estructura de datos elegida para tal fin, en este caso los tries..

La realización de este proyecto tratará de modelar una serie de patrones melódicos a partir de una colección de partituras en formato MusicXML previamente seleccionadas y clasificadas por géneros, de las cuales se extraerá la información necesaria para crear melodías que sigan los patrones extraídos.

Aunque no es el objetivo del proyecto, la composición automática de melodías está ligada a la cuestión sobre si las máquinas pueden conseguir creatividad comparable a la de un ser humano. La música es un medio particularmente adecuado para el estudio de esta cuestión ya que a diferencia de otras formas artísticas como la pintura, utiliza una representación simbólica, el pentagrama, para su transmisión e interpretación.

En cuanto al enfoque del proyecto, cabe reseñar que el proyecto no tiene ninguna restricción de género musical a la hora de extraer sus patrones musicales. Por último, el proyecto ha sido desarrollado de manera modular, de modo que cada parte pueda ser fácilmente modificada debido a la inclusión de nuevos requisitos y funcionalidades.

1.2 Objetivos

El objetivo principal del proyecto es la composición de melodías a partir de patrones melódicos obtenidos del modelado de un estilo musical particular. En función de este objetivo principal se proponen los siguientes subobjetivos, cuyo cumplimiento permitirá alcanzar el objetivo principal.

- Subobjetivo 1: Construcción de la Base de datos MusicXML.

El primer subobjetivo a completar es el recopilar las partituras de MusicXML necesarias para el proyecto, ya que estas partituras contienen la información musical necesaria para el modelado de patrones. Para ello se escogen 3 estilos musicales concretos (Clásica, Metal, Jazz). Una vez escogidos, se elegirán partituras de MusicXML de cada estilo de modo que haya una muestra significativa de cada uno.

- Subobjetivo 2: Desarrollo de una librería para el manejo de tries.

El siguiente subobjetivo es crear una librería para manejar la estructura de datos trie. En esta estructura se almacenará y procesará la información extraída de las partituras, implementando las operaciones necesarias para este tratamiento.

- Subobjetivo 3: Extracción del conjunto de patrones melódicos.

Para cumplir este subobjetivo, se debe desarrollar un software que permita extraer la información de las partituras del estilo musical elegido con el fin de obtener las pautas que sigue este género. A partir de los patrones extraídos podremos generar melodías de un estilo concreto.

- Subobjetivo 4: Composición de melodía con los patrones melódicos extraídos.

Una vez que se tienen los patrones melódicos, el siguiente subobjetivo es componer una melodía en formato MIDI basándose en los patrones extraídos y en la secuencia de concatenación de los mismos, indicada por el usuario.

- Subobjetivo 5: Implementación de Interfaz gráfica.

Por último, el subobjetivo final del desarrollo de la aplicación es el diseño e implementación de una interfaz gráfica. De este modo, se obtendrá una interfaz que facilite al usuario la elección del estilo musical a tratar, la ubicación del directorio que contienen los MusicXML del estilo escogido, y la secuencia de patrones que compondrá la melodía.

1.3 Fases del desarrollo

Este apartado describe las fases que se han seguido durante el desarrollo del proyecto.

1. Fases de implementación

Para alcanzar los objetivos planteados han sido necesarias las siguientes fases de implementación:

- Fase 1.1 Desarrollo de una librería para el manejo de tries.

En esta primera fase se realiza el desarrollo de la librería para el manejo de tries, el cual incluye toda la funcionalidad para manejar esta estructura. De este modo, toda la información que contienen las partituras de MusicXML será recogida en los tries.

Para ello se debe crear una librería que contenga la definición de esta estructura y todas las operaciones necesarias para este proceso. La estructura del trie guardará las transiciones de intervalos (tanto de altura como de duración) así como el número de veces que se repite cada intervalo.

Con la creación de esta librería nos aseguramos de tener las herramientas necesarias para tratar toda la información procedente de la extracción de información de las partituras MusicXML.

- Fase 1.2 Extracción del conjunto de patrones melódicos.

En esta fase se programa toda la funcionalidad de extracción de información de las partituras MusicXML y la posterior composición a partir de ellas.

El primer cometido es lograr sacar de las partituras la información en forma de intervalos y frases que serán usadas posteriormente. Estas frases son guardadas en una clase especialmente creada para ello.

A partir de aquí, en esta fase se han implementado las funciones necesarias para generar las melodías a partir de la información extraída de las partituras. Con la creación de todas estas funciones se tienen las herramientas necesarias para poder extraer la información de las partituras y crear nuevas melodías a partir de ellas.

- Fase 1.3 Implementación de Interfaz gráfica.

Mediante esta última fase de implementación se dota al sistema de una interfaz gráfica. Esta interfaz servirá de guía para el usuario final de la aplicación, indicándole los parámetros de entrada que necesita el programa, tales como el estilo a modelar y la secuencia de patrones que compondrá la melodía resultante.

2. Fase de Experimentación

Para alcanzar los objetivos planteados han sido necesarias las siguientes fases de experimentación:

- Fase 2.1 Construcción de la Base de datos MusicXML.

El primer paso a dar es recopilar partituras de MusicXML del estilo musical que se quiere modelar. Para el desarrollo de este proyecto se han escogido los estilos Metal, Clásica y Blues, no obstante el objetivo es que se pueda modelar cualquier otro estilo.

Para que el análisis del estilo sea significativo, es necesario reunir una muestra suficiente de partituras. En este caso el estilo escogido es música Clásica y se va a contar con una muestra de unas 10 partituras, de modo que el resultado del modelado sea lo más significativo posible.

A continuación, se crea un directorio donde se guardan todas las partituras correspondientes a este estilo. La aplicación, al inicio de la ejecución, lee de este directorio todas las partituras que estén aquí ubicadas.

- Fase 2.2 Extracción de patrones y generación de melodías.

En esta fase se trata de extraer los patrones melódicos de la información obtenida de las partituras en MusicXML. El sistema desarrollado leerá de forma iterativa las partituras contenidas, extrayendo la información que resulta necesaria para el proyecto.

La última fase para completar la ejecución del sistema es crear una melodía a partir de los patrones generados. Para ello el usuario introducirá una lista de patrones que componen la melodía.

A cada patrón generado se le asignará un identificador. La concatenación de estos patrones de un estilo determinado forma la melodía resultante. Una vez generada la melodía final, se procede a generar los acordes que le servirán de acompañamiento.

- Fase 2.3 Evaluación de las melodías generadas.

Una vez tenemos generado un conjunto significativo de melodías para los distintos estilos, en esta fase las evaluamos analizando y comparando su estructura.

1.4 Estructura de la memoria

Para facilitar la lectura de la memoria, se incluye a continuación un breve resumen de cada capítulo:

- Capítulo 1 Introducción y objetivos: En este capítulo se incluye una breve introducción sobre el trabajo realizado, así como los objetivos a cumplir. Por último se tratan las fases de desarrollo de las que ha constado el proyecto
- Capítulo 2 Medios empleados: En este apartado se incluye la descripción de los conocimientos básicos y las herramientas para llevar cabo el proyecto.
- Capítulo 3 Estado del arte: Este capítulo repasa los sistemas que utilizan Inteligencia Artificial para la composición automática de música.
- Capítulo 4 Planificación y presupuesto: En este capítulo se incluye el presupuesto establecido para la elaboración del presupuesto, así como la planificación seguida a lo largo del proceso de desarrollo.
- Capítulo 5 Trabajo realizado: En este capítulo se tratan todos los aspectos relacionados con la generación de código y algoritmos empleados para elaborar el proyecto. Se incluye el flujo general del sistema y la descripción de todos los módulos creados en el proyecto.
- Capítulo 6 Resultados experimentales: Incluye la descripción de todos los experimentos realizados y los preparativos necesarios para realizarlos. También se valoran los resultados obtenidos según el estilo musical tratado, clásica, metal o blues.
- Capítulo 7 Conclusiones: En este capítulo se incluye las conclusiones finales tras realizar todos los experimentos, además de posibles líneas de desarrollo futuras del proyecto.

2 Medios empleados

Con el fin de facilitar el seguimiento del documento, este apartado explica nociones básicas de los elementos utilizados en el desarrollo del proyecto. Primero, explica conceptos de teoría musical. Segundo explica las características del formato MIDI. Tercero hace referencia a MusicXML, formato en el que están creadas las partituras que se utilizan como entrada en el proyecto, el lenguaje Java y las librerías utilizadas para el desarrollo del proyecto. Finalmente introduce la estructura de datos trie, utilizada en el proyecto.

2.1 Teoría Musical

La música, según las definiciones más clásicas, es un conjunto de sonidos y silencios organizados de manera vertical y horizontal. A esta verticalidad y horizontalidad las llamamos respectivamente armonía y melodía. Dentro del trabajo que se realiza en el proyecto, cabe decir que está centrado en la organización horizontal, es decir, la melodía.



Figura 1 Organización vertical y horizontal [Ltm]

En la figura 1 podemos ver en la parte inferior un ejemplo de organización vertical, la cual se refiere a la composición de acordes y acompañamientos. A su vez, en el pentagrama de la parte superior tenemos un ejemplo de organización horizontal, la cual se refiere a la composición de melodías.

Parámetros del sonido

El sonido podemos definirlo como la sensación que percibe el oído al recibir variaciones de presión generadas por el movimiento vibratorio de un cuerpo sonoro. Es transmitido por el medio que rodea al cuerpo sonoro, generalmente el aire de la atmósfera. A la ausencia perceptible de sonido la llamamos silencio, aunque no sea totalmente cierto, ya que el silencio absoluto no puede darse en la naturaleza.

A continuación vamos a definir sus cuatro parámetros fundamentales.

Altura

La altura es el resultado de la frecuencia que produce un cuerpo sonoro. Clasificando estas frecuencias podemos dar nombres a las notas, las cuales tienen una longitud de onda característica y única. Esta cualidad nos permite diferenciar entre tonos graves y agudos. Para representarla se utilizan los valores de nota y octava, ya que una misma nota tocada en diferentes octavas tiene una altura distinta.

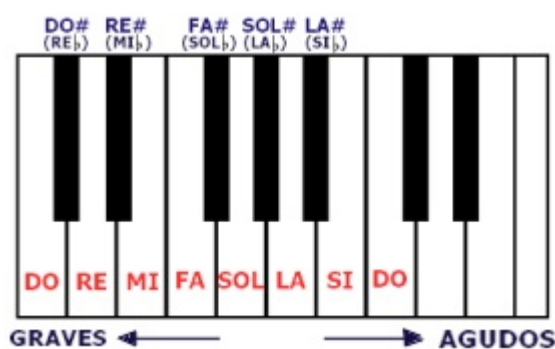


Figura 2 Altura [Wik]

En la figura 2 observamos las 12 notas (DO, DO#,...) que componen una octava.

Duración

La duración corresponde al tiempo que duran las vibraciones que produce un sonido. Este atributo está relacionado con el ritmo.

La duración de los sonidos se representa por medio de la figura de las notas. La figura que representa la unidad es la redonda, y sirve como punto de referencia para conocer el valor del resto de las figuras. Hay que aclarar que en el lenguaje musical, "valor" equivale a duración de un sonido. Los valores de estas notas se subdividen en tal forma que cada una de ellas vale la mitad que el valor anterior.

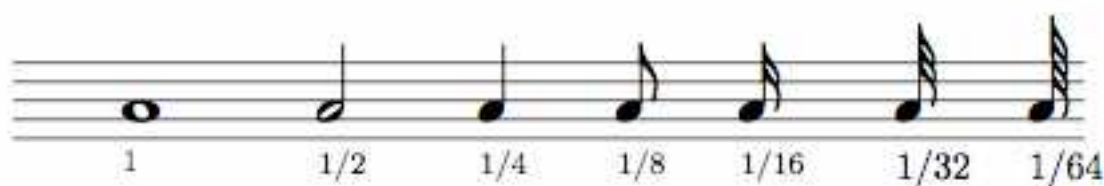


Figura 3 Duración

En la figura 3 se muestran los símbolos correspondientes para las duraciones de redonda, blanca, negra, corchea, semicorchea, fusa y semifusa.

Intensidad

La intensidad es la fuerza con la que se produce un sonido, representada por una amplitud de onda. Es lo que comúnmente entendemos por volumen.

Timbre

El timbre (o forma de onda) es la cualidad del sonido que nos permite distinguir diferentes instrumentos o voces aunque estén produciendo sonidos con la misma altura, duración e intensidad.

Elementos de la música

La organización coherente de los sonidos y los silencios nos da los parámetros fundamentales de la música, que son la melodía, la armonía y el ritmo. A continuación vamos a definirlos.

Melodía

Podemos definir la melodía como un conjunto de sonidos, dentro de un contexto sonoro particular, que suenan sucesivamente uno después de otro y que se perciben con identidad y sentido propio. Dentro de la estructura de la melodía también se incluyen los silencios, los cuales ponen pausas al discurso melódico.

Métrica

La métrica es la pauta de repetición a intervalos regulares, y en ciertas ocasiones irregulares, de sonidos fuertes o débiles y silencios en una composición.

Ritmo

El ritmo, es el resultado final de los elementos anteriores, a veces con variaciones muy notorias, pero en una muy general apreciación se trata de la capacidad de generar contraste en la música, esto es provocado por las diferentes dinámicas, timbres, texturas y sonidos.

Armonía

La armonía regula la concordancia entre sonidos que suenan simultáneamente y su enlace con sonidos vecinos. Su unidad básica es el acorde.

Acorde

En música y teoría musical, un acorde consiste en un conjunto de dos o más notas diferentes que suenan simultáneamente o en sucesión y que constituyen una unidad armónica dentro de la composición. En determinados contextos, un acorde también puede ser percibido como tal aunque no suenen todas sus notas. Pueden formarse acordes con las notas de un mismo instrumento o con notas de diferentes instrumentos (incluyendo la voz humana) tocados a la vez. La distancia entre dos notas musicales se conoce como intervalo musical; los intervalos musicales, combinados, determinan los diferentes tipos de acordes. Cada tipo de acorde puede presentar como tono fundamental cualquiera de las doce notas musicales (do, do#, re, mi b, mi, fa, fa#, sol, la b, la, si b, si). Este tono

fundamental (también conocido como nota fundamental, nota tónica, fundamental o tónica) determina la tonalidad del acorde y constituye la referencia para los intervalos del mismo.

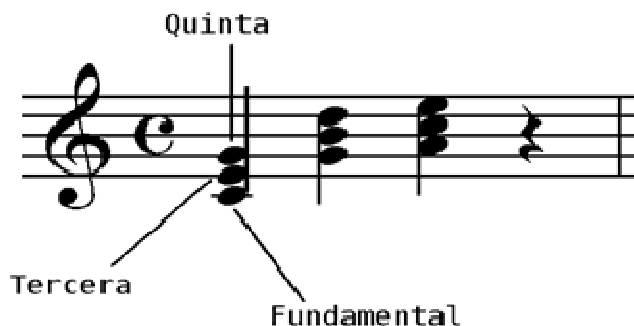


Figura 4 Acorde fundamental [Sax]

En la figura 4 podemos ver un ejemplo de acorde fundamental donde la nota do ejerce de tónica, siendo la nota mi su tercera mayor y la nota sol su quinta justa.

Intervalos Musicales

Un intervalo es la diferencia de altura entre 2 notas musicales, medida cuantitativamente en grados o notas naturales y cualitativamente en tonos y semitonos.

A continuación se muestra un ejemplo:

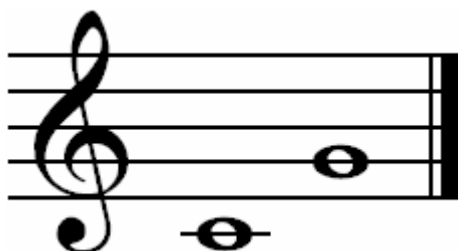


Figura 5 Intervalo musical [Wik]

Este es un intervalo melódico de quinta justa ascendente, partiendo de Do. Recibe el nombre de *quinta* porque hay una distancia de cinco grados entre las notas que lo forman (do y sol). El calificativo de "justa" viene porque no se diferencia entre escalas menores y mayores.

Tipos de Intervalos Musicales

Es posible analizar y clasificar los intervalos, tan solo tenemos que colocar el número correspondiente a la distancia entre las notas y después identificarlo dentro de su clasificación. A continuación se muestra la clasificación de estos intervalos:

Tipo	Distancia entre notas
2ª Menor	½ tono
2ª Mayor	1 tono
3ª Menor	1 tono y ½
3ª Mayor	2 tonos
4ª Justa	2 tonos y ½
4ª Aumentada	3 tonos
5ª Disminuida	3 tonos
5ª Justa	3 tonos y ½
5ª Aumentada	4 tonos
6ª Menor	4 tonos
6ª Mayor	4 tonos y ½
7ª Menor	5 tonos
7ª Mayor	5 tonos y ½
8ª Justa	6 tonos

Tabla 1 Tipos de intervalos musicales

2.2 MIDI

MIDI (Musical Instrument Digital Interface) es un formato electrónico desarrollado en la época de los 70 para expresar en términos digitales eventos musicales. Desde su desarrollo hasta la fecha, no ha sufrido modificaciones mayores. Cualquier proyecto musical abordado en la actualidad en un plano informático, está sin duda, directa o indirectamente influenciado por su especificación. De igual manera, compatibilizar con el formato es garantizar una integración con el gran abanico de aplicaciones musicales que existe en la actualidad.

Formato MIDI

El formato MIDI permite convertir la información de una obra musical en datos digitales. Esta información es transmitida a través de una secuencia de instrucciones, llamados mensajes MIDI, que indican a un dispositivo receptor como debe interpretar esa secuencia musical. Este dispositivo receptor es el encargado de generar en tiempo real el sonido correspondiente a la secuencia.

El protocolo MIDI también incluye un hardware consistente en un grupo estándar de conectores denominados In, Out y Thru.

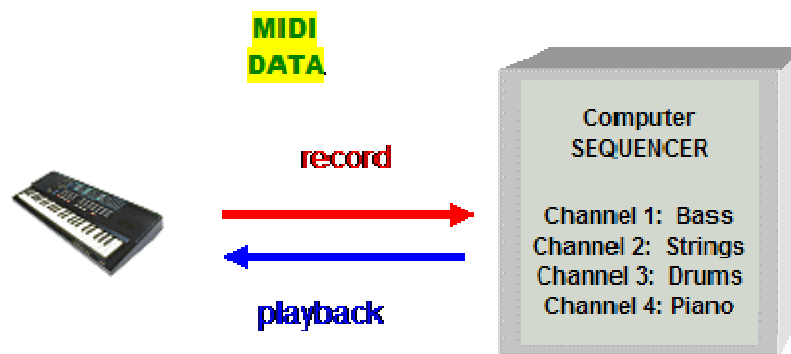


Figura 6 Flujo de datos MIDI [Css]

Normalmente los dispositivos MIDI son capaces de enviar y recibir información, pero desempeñan un papel diferente dependiendo de si están recibiendo o enviando información. La figura 6 muestra el flujo de datos de un dispositivo MIDI en grabación y reproducción de secuencias MIDI.

El flujo de datos MIDI se compone de una serie de bits unidireccional y asíncrona que es transmitida a una velocidad de 31,25 Kbits/seg. Este flujo de datos es emitido a partir de un controlador, que puede ser por ejemplo un teclado o un secuenciador, a tiempo real y a través del conector de MIDI Out.

El dispositivo receptor de los datos, a través de su conector MIDI In, responde a los mensajes y emite el sonido mediante sus salidas de audio. Se debe tener en cuenta que la mayoría de los teclados MIDI incorporan a la vez el controlador y el sintetizador de sonidos, por lo que existe un enlace interno entre ellos. Este enlace puede estar activado o no en función de si se está utilizando o no el teclado MIDI con un secuenciador externo.

El puerto MIDI físico tiene capacidad para alojar hasta 16 canales MIDI independientes gracias al “Número de canal”, un parámetro de 4 bits que permite diferenciar cada canal. Un teclado, generalmente, se puede configurar para emitir por cualquiera de estos 16 canales, mientras que un generador de sonido puede configurarse para recibir información en un canal o canales específicos.

La información que recibe el conector MIDI In de un dispositivo es retransmitida mediante el conector MIDI Thru, de modo que es posible conectar en cadena varios dispositivos y que el flujo de datos llegue a todos por igual. De esta manera, se puede utilizar un teclado controlador para introducir los datos en un secuenciador que a su vez se encarga, a través de su conector MIDI Out, de enviarlos hacia distintos módulos generadores de sonidos. Así es posible crear música compuesta por distintas partes instrumentales y que cada una de ellas pueda ser “reproducida” por un instrumento distinto. Cada una de las partes se reproducirá en un canal MIDI distinto, que se corresponde con el canal de recepción del módulo que disponga del instrumento elegido.

Mensajes MIDI

Al trabajar con MIDI se generan unos mensajes compuestos de 1 byte de estado de 8 bits seguido de uno o dos bytes de datos. Dentro de estos mensajes se distinguen entre mensajes de canal, los cuales se dirigen a un canal concreto que se indica en su byte de estado, y los mensajes de sistema, que como su nombre indica no se dirigen a ningún canal en concreto, recibiendo el mensaje todos los dispositivos conectados. A continuación se muestra una tabla con los mensajes MIDI más habituales, además de una breve explicación de los mensajes más importantes:

Byte Estado	Descripción
1000cccc	Desactivación de la nota
1001cccc	Activación de la nota
1010cccc	PostPulsación polifónica
1011cccc	Cambio de Control
1100cccc	Cambio de programa
1101cccc	Postpulsación monofónica de canal
1110cccc	Pitch
11110000	Mensaje exclusivo del fabricante
11110001	Mensaje de trama temporal
11110010	Puntero posición de canción
11110011	Selección de canción
11110110	Requerimiento de entonación
11110111	Fin de mensaje exclusivo
11111000	Reloj de Temporización
11111010	Inicio
11111011	Continuación
11111100	Parada
11111110	Espera Activa
11111111	Reseteo del Sistema

Tabla 2 Tipos de mensajes MIDI

Los primeros bytes, cuyos últimos cuatro bits están marcados como "cccc", se refieren a mensajes de canal; el resto de bytes son mensajes de sistema.

Principales mensajes MIDI:

- Activación y desactivación de la nota: cuando se pulsa una nota, se transmite la información de qué canal MIDI se está usando, qué nota fue pulsada, y con qué fuerza (velocidad MIDI). Al soltarla, se envía un mensaje indicando el canal MIDI y el número de la nota liberada.
- Postpulsación: También denominada presión, ya que sirve para expresar la mayor o menor presión aplicada sobre las teclas después de haberlas pulsado

inicialmente. Se usa para que el instrumentista pueda variar a su elección el volumen de la nota mientras la interpreta.

- Cambio de programa: Se refiere a la posibilidad de indicar a cada parte del sintetizador qué sonido se le desea asignar a partir de ahora. Así pues, una composición no está limitada a 16 instrumentos, aunque ése sea el límite de instrumentos simultáneos.
- Cambio de control: MIDI también permite transmitir información sobre la forma de la interpretación, así como datos adicionales. Existen 128 parámetros de control (controladores), y cada uno de ellos puede adoptar un valor de 0 a 127. Por ejemplo, el controlador 7 es el volumen, independiente para cada canal MIDI, al igual que el número 10 (panorama estéreo). Con estos dos podemos ajustar a nuestro gusto la mezcla final de todos los sonidos, al igual que lo haríamos con una mesa de mezclas. Otros controladores sirven para indicar el uso de los diferentes pedales del piano, datos sobre el soplo en instrumentos de viento, etc.
- Pitch: Sirve para desafinar el sonido, simulando así el estiramiento de las cuerdas de una guitarra, o similar.

2.3 MusicXML

Mientras que en el plano musical los programas informáticos se entienden usando el estándar MIDI, no existe manera de expresar en este estándar información sobre las notaciones o su colocación en la partitura de la pieza. En este marco se han desarrollado varios intentos de paliar estas limitaciones de representación del MIDI.

Estos nuevos estándares no intentan sustituir MIDI como formato de representación musical. Su intención es la de crear un estándar que se encargue de asegurar una representación de la notación de esa música de manera homogénea entre los distintos programas.

De las numerosas representaciones que existen, varias tienen en común el uso de XML como medio expresivo. Algunas de estas son WEDELMUSIC, MUTATED, XCHORDS y por último MusicXML.

Esta última iniciativa es la más madura, principalmente por haber sido desarrollada por una empresa comercial conjunta y paralelamente con un software de uso extendido. A la par las licencias son libres lo cual asegura que surgirá software libre de terceras partes que soporte este formato. Otra razón importante para su adopción en primer lugar es que su estructuración de la información musical es exactamente la misma que adoptamos desde un inicio en este proyecto.

MusicXML es un formato de ficheros de música basado en XML. Es un proyecto abierto. Se ha desarrollado por Recordare. El estándar MusicXML está definido por una serie de definiciones de tipo de documento (DTD) libremente redistribuibles bajo la licencia pública de MusicXML DTD.

Las principales metas de MusicXML son proponer un estándar de intercambio y representación de archivos musicales Internet-friendly y que no fuese en formato binario. Estas dos características apuntaban a la posible aplicación de XML como plantilla. La adopción de XML significa crear un formato con un alto nivel de legibilidad humana y con herramientas de apoyo existentes.

MusicXML está basado en las normas y formatos definidos en el meta-lenguaje XML. XML es un subconjunto de SGML (Standard Generalized Markup Language). Los documentos XML son contenedores de información, MusicXML ha sido diseñado para la representación de música en partitura. Prevé el almacenamiento digital, así como el intercambio de los datos musicales.

Estructura

Esta sección cubre los principales elementos que están definidos en el estándar. Cada fichero xml está dividido inevitablemente en dos partes. Por un lado, dentro del tag <PART-LIST> encontramos un resumen de las partes siguientes que contendrán la notación. Esto es necesario para introducir los elementos que más tarde serán leídos.

A continuación de este resumen, vamos a encontrar las partes. Cada una va rodeada de los tags <PART> e identificada por un id unívoco. Para cada parte, encontramos todos los compases enunciados como medidas (<measures>). Están numeradas con el atributo xml number y esto permite ordenarlas en el tiempo y relacionar varias partes entre ellas.

Encontramos aquí dentro ya elementos relevantes para la maquetación como la clave (<clef>) que nos indica en que clave es interpretada la partitura. Por otra parte dentro de <attributes> tenemos los siguientes tags dándonos información relevante:

- <division>. Dictamina en cuantas porciones de tiempo (ver debajo) dividimos las duraciones.
- <time>. Es el compás. <beats> entre <beats-type>

Salimos de la definición de atributos y encontramos ya los elementos musicales. Todo está expresado como notas. El tag <note> tiene anidado dentro los siguientes tags de los cuales extraeremos la información musical de cada una de las composiciones:

- <pitch>. La altura de la nota.
- <duration>. La duración de la nota expresada en función de divisions.
- <type>. Información meramente maquetadora sobre el carácter que representará la nota.

Ejemplo

A continuación se muestra un ejemplo de partitura MusicXML que contiene los elementos previamente descritos.

```
<score-partwise version="2.0">
  <part-list>
    <score-part id="P1">
      <part-name>Music</part-name>
    </score-part>
  </part-list>
  <part id="P1">
    <measure number="1">
      <attributes>
        <divisions>1</divisions>
        <key>
          <fifths>0</fifths>
        </key>
        <time>
          <beats>4</beats>
          <beat-type>4</beat-type>
        </time>
        <clef>
          <sign>G</sign>
          <line>2</line>
        </clef>
      </attributes>
      <note>
        <pitch>
          <step>C</step>
          <octave>4</octave>
        </pitch>
        <duration>4</duration>
        <type>whole</type>
      </note>
    </measure>
  </part>
</score-partwise>
```

Figura 7 Ejemplo MusicXML [Wik]

En el ejemplo podemos ver una sola nota envuelta en una serie de tags. Primero se muestra el tag “<measure number=1>” que nos indica que es el primer compás de la partitura. A continuación tenemos una serie de tags que nos indican el contenido de este compás, como el tag “<beats>4</beats>”, que nos indica que el compás es un 4/4, y el tag “<clef>”, dentro del cual se nos indica que está en clave de Sol. Después de los tags previamente nombrados, tenemos la definición de la nota que compone el compás. Dentro del tag “<pitch>” tenemos la definición de la nota y su octava correspondiente, siendo un Do en la octava 4 en este caso. Para finalizar, tenemos los tags “<duration>” y

“<type>”, los cuales definen la duración de la nota y el tipo, siendo la duración 4 y el tipo una redonda en este caso.

2.4 Java

Java es un lenguaje de programación orientado a objetos de código interpretado. Esto lo hace multiplataforma, portable sin recompilación. Ha sido el lenguaje elegido para generar todo el código del que consta el proyecto.

En esta sección explicaré el API Java Sound, utilizado para generar los MIDI que se obtienen como salida de la aplicación, el paquete Player y el paquete Swing, utilizado para crear la interfaz.

2.4.1 API Java Sound

Java Sound [Jav] es un API de bajo nivel para el tratamiento de efectos y el control de entrada y salida de sonido. Este API incluye soporte tanto para audio digital como para MIDI, componiéndose de cuatro paquetes principales:

- javax.sound.sampled
- javax.sound.midi
- javax.sound.sampled.spi
- javax.sound.midi.spi

El paquete **javax.sound.sampled** contiene las clases necesarias para el manejo del sonido muestreado y el paquete javax.sound.midi proporciona la interfaz para la síntesis, secuenciamiento y transporte de eventos MIDI, aportando los paquetes javax.sound.sampled.spi y javax.sound.midi.spi la interfaz para los desarrolladores de servicios sobre las anteriores.

Para la realización de este proyecto, se ha utilizado el paquete **javax.sound.midi**. Una de las clases abstractas de este paquete es MidiMessage. Esta clase abstracta representa un mensaje definido por el estándar de especificación de archivos MIDI. Los ShortMessages son los mensajes más comunes.

La clase MidiEvent representa eventos que deben ser almacenados en un fichero MIDI. El API incluye métodos para manejar eventos en el tiempo. Los eventos MIDI se organizan en pistas.

Cada nota de una composición musical se representa en una pista mediante, al menos, dos eventos: Note On para indicar el comienzo de la nota y Note Off para indicar el final. La clase Track representa una colección de MidiEvents. De igual manera, la clase Sequence representa una colección de Tracks.

Un secuenciador es un dispositivo para capturar y reproducir secuencias de eventos MIDI. La interfaz Sequencer añade a su superinterface, MidiDevice, métodos para las operaciones básicas de secuenciación MIDI.

Un sintetizador es un dispositivo para la generación de sonido, lo cual ocurre cuando el MidiChannel de un Synthesizer recibe un mensaje. El MidiChannel utiliza la información de la nota en estos mensajes para sintetizar la música. Sin embargo, la información de la nota es insuficiente; el sintetizador requiere también instrucciones precisas sobre cómo crear la señal de audio para cada nota.

La interfaz Synthesizer incluye métodos para cargar y descargar instrumentos desde un banco de sonidos que emulan instrumentos tradicionales o efectos de sonido. Existen diferentes bancos de sonido, cada uno representados con un número, y, dentro de ellos, los instrumentos están organizados por número de programa. Al cargar un instrumento, éste se dispone para la síntesis de notas. Para que el instrumento toque las notas, el MidiChannel debe recibir un mensaje de cambio de programa, que hace que el banco de sonidos y el número de programa del instrumento que se seleccione.

2.4.2 Paquete Player

Este paquete está basado en el API Java Sound [Jav] previamente mencionado. Fue desarrollado para un PFC anterior que también utilizaba MIDI, por lo que será reutilizado en este proyecto, dado su utilidad para la tarea que nos ocupa.

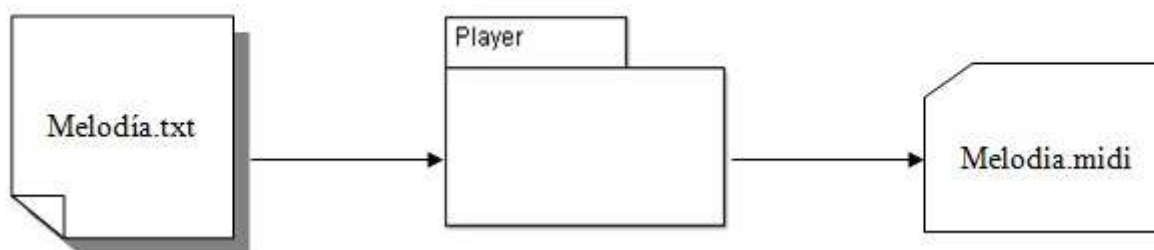


Figura 8 Entradas y salidas del paquete Player

Como se puede ver en la figura 8, el paquete Player tiene como entrada un archivo de extensión .txt en el cual está incluida toda la información musical de la melodía a reproducir. Como salida genera un archivo de extensión MIDI que es la representación en este formato del .txt de entrada.

A continuación se detallan las clases que componen este paquete.

Clase NewPlayer

La clase **NewPlayer** es la clase principal del módulo Player. Su método principal se encarga de crear los objetos Track, Sequence, Sequencer y Synthesizer con sus propiedades.

Una vez leído e interpretado el fichero de texto, se genera un archivo con el mismo nombre que el fichero interpretado y extensión .midi, el cual contiene la melodía lista para su reproducción.

El método interpretarFichero se encarga de leer secuencialmente el fichero de texto que se recibe como parámetro. Dentro de cada línea identifica el nombre del instrumento, su volumen, su timbre y su partitura, y llama a los métodos tocar y tocarPercusion de la clase Instrumento para interpretar la partitura de los instrumentos melódicos y de percusión, respectivamente.

Clase Instrumento

La clase Instrumento tiene una doble definición para su constructor, dependiendo de si el instrumento es melódico o de percusión. Al instanciar un objeto de tipo Instrumento, se crea un evento MIDI mediante el cual se emite un ShortMessage para cambiar el programa.

Los métodos tocar y tocarPercusion reciben como parámetro de entrada el fragmento de línea correspondiente a la partitura de un instrumento melódico o de percusión, respectivamente. Dicho fragmento se va examinando carácter a carácter, y con los caracteres leídos se van calculando las alturas y duraciones de cada nota de la partitura, entendiéndose el espacio como un separador entre dos notas o acordes contiguos. El espacio es el carácter que indica que se debe generar un nuevo evento MIDI, y para ello se hace una llamada al método tocarNota, encargado de tal fin.

El método tocarNota recibe como parámetro el track, la posición inicial, la altura, la duración de la nota y el volumen. Con estos valores, genera un mensaje NOTE_ON y otro mensaje NOTE_OFF indicando el inicio y el fin de una nota dentro del track.

2.4.3 Paquete Swing

El paquete javax.Swing[Swi] forma parte de la JFC (Java Foundation Classes) de la plataforma Java y su principal función es facilitar la elaboración de interfaces de usuario. Para ello este paquete ofrece al programador una serie de componentes predefinidos como botones, tablas, marcos, etc.

En esta sección se explicará la arquitectura de este paquete, así como los componentes más relevantes utilizados en la elaboración del interfaz de usuario de la aplicación.

Arquitectura

El paquete javax.Swing[Swi] es un Framework MVC (Modelo Vista-Controlador) que permite desarrollar interfaces de usuario con independencia de la plataforma en la que se trabaje. El Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. A continuación vemos cada componente del modelo con más detenimiento:

- **Modelo:** Esta es la representación específica de la información con la cual el sistema opera. En resumen, el modelo se limita a lo relativo de la vista y su controlador facilitando las presentaciones visuales complejas.
- **Vista:** Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador:** Este responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y a la vista.

Aquí mostramos un diagrama de este modelo:

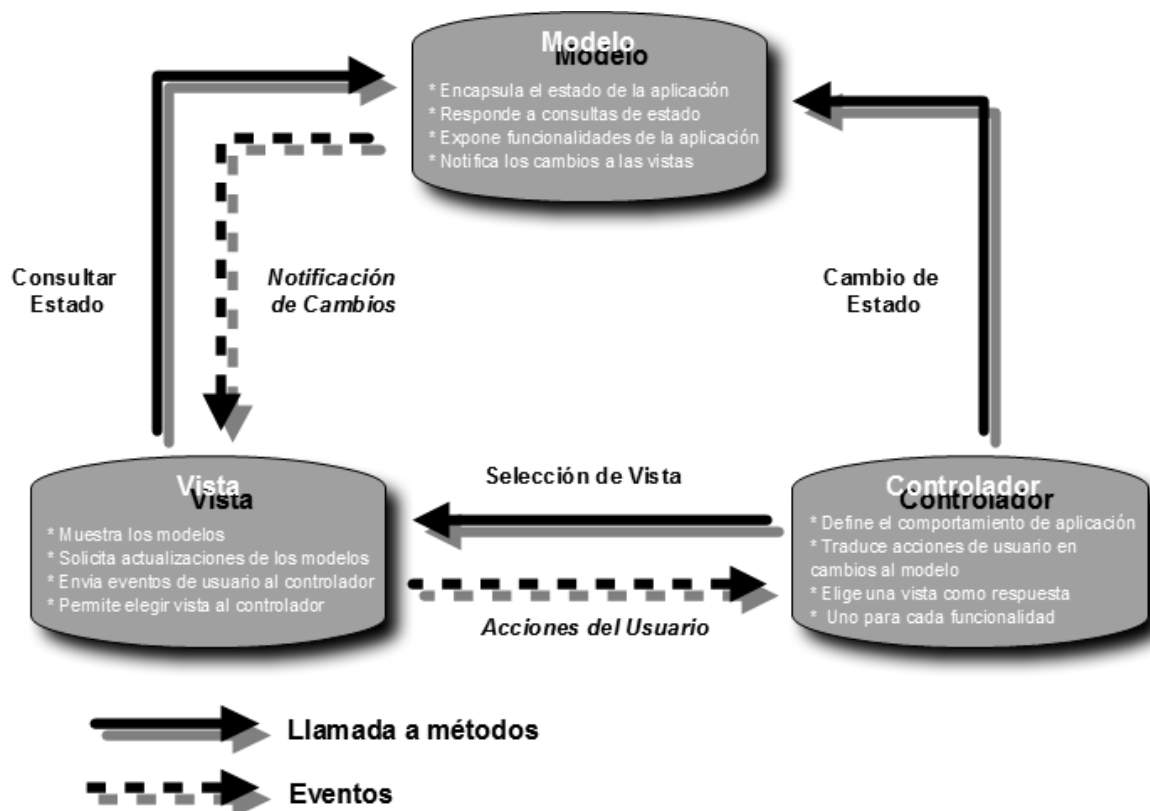


Figura 9 Modelo Vista-Controlador [Dav]

El flujo de ejecución de este modelo es el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.).
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario y el controlador gestiona el evento que llega.
3. El controlador accede al modelo, actualizándolo o modificándolo de forma adecuada a la acción solicitada por él.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada.

para el usuario donde se refleja los cambios en el modelo (por ejemplo, actualiza el perfil del usuario).

La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

Componentes de Swing

En este apartado vamos a ver los componentes del paquete Swing que han sido utilizados para desarrollar la interfaz de usuario del proyecto. Estos componentes se reconocen por su nomenclatura, ya que empiezan por "J" más el nombre del componente (JButton por ejemplo) y generan código que controla los eventos asociados a cada uno de ellos.

- JFileChooser: permite mostrar una ventana para que el usuario navegue por los directorios y elija un fichero.

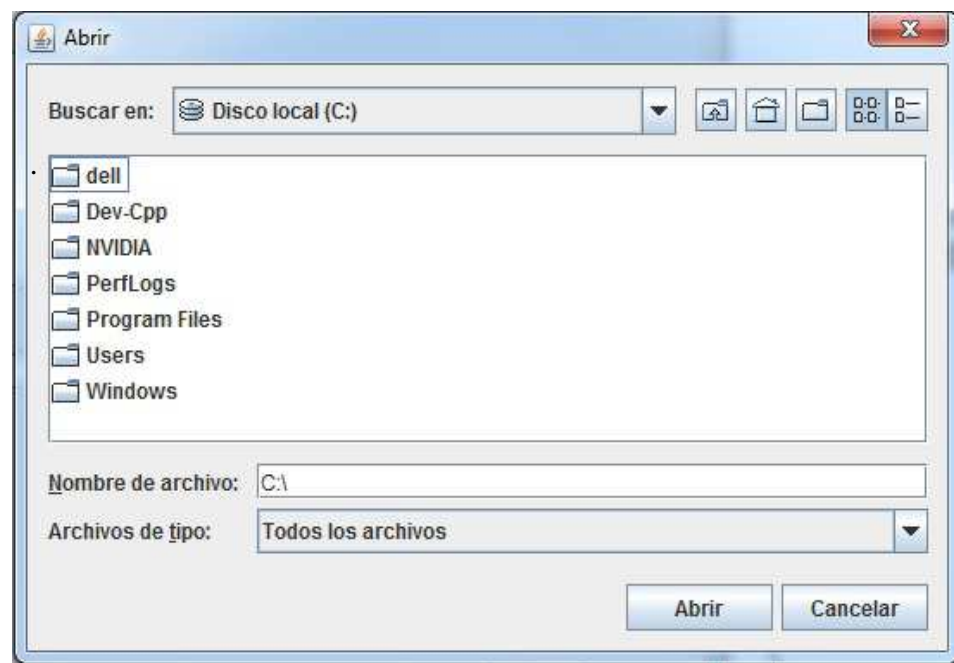


Figura 10 Ejemplo JFileChooser

- JTextField: permite al usuario escribir una cantidad limitada de texto en el recuadro habilitado para ello. Cuando el usuario indica que la entrada está completa (generalmente cuando se presiona la tecla enter), el Textfield dispara el evento asociado, que será tratado según se haya programado.



Figura 11 Ejemplo JTextField

- JButton: permite a un usuario crear un botón con una acción asociada. Cuando el usuario hace clic sobre este botón se dispara el evento asociado a este componente.



Figura 12 Ejemplo JButton

- JList: Este componente presenta al usuario un grupo de elementos dispuestos en una lista de una o varias columnas. El usuario puede elegir elementos de estas listas mediante clics de ratón, que dispararán los eventos asociados.

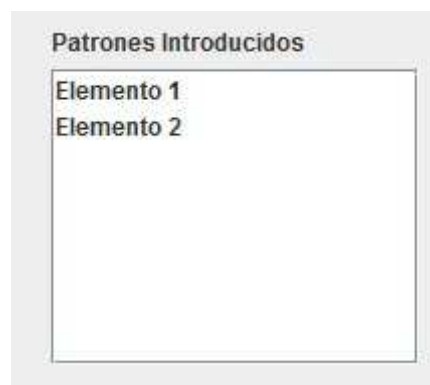


Figura 13 Ejemplo JList

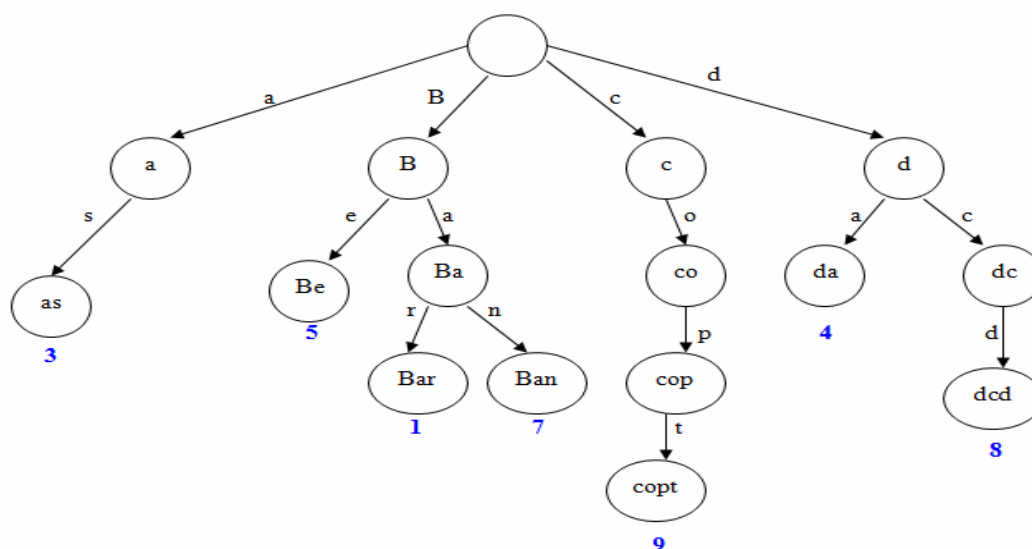
2.5 Tries

Una estructura trie es esencialmente un árbol n-ario, dónde n es el factor de ramificación máximo de los nodos del árbol. Principalmente, los trie son árboles de búsqueda, donde las claves de búsqueda son interpretadas partidas en porciones de claves. Estas porciones son utilizadas para armar la estructura de datos de búsqueda.

Los tries están formados por dos tipos de nodos:

- Nodo Interno: Estos nodos contienen un número fijo de entradas, donde cada entrada esta compuesta por un indicador de tipo de nodo y el enlace ha dicho nodo.
- Nodo Hoja: Cuando un subtrie contiene solamente una valor de clave, este apunta a un Nodo Hoja. El Nodo Hoja contiene el valor de la clave y toda información relevante de ese valor de clave.

Veamos un ejemplo de trie:



En este trie podemos observar los nodos hoja *as*, *Be*, *Bar*, *Ban*, *copt*, *da*, *dcd* y el camino seguido hasta llegar a ellos, así como en color azul el número de apariciones de cada nodo hoja.

Las principales ventajas de los tries respecto a los tradicionales Árboles de búsqueda binaria son las siguientes:

- 32/110

Por otro lado, la estructura trie puede utilizarse para sustituir otras estructuras de datos como las tablas hash, presentando las siguientes ventajas y desventajas:

Las principales ventajas de un trie respecto a una tabla hash son:

- En un trie no se producen colisiones de claves.
- No es necesario definir una función de hash o modificarla si hay más claves.
- El tiempo de búsqueda en un trie es del orden $O(l)$, siendo este tiempo en una tabla hash imperfecta del orden de $O(n)$ debido a las colisiones de claves.
- Un trie puede proporcionarnos un ordenamiento alfabético de las entradas por clave.
- Los contenedores que almacenan distintos valores asociados a una única clave sólo son necesarios si tenemos más de un valor asociada a una única clave. En una tabla hash siempre se necesitan estos contenedores para las colisiones de clave.

Las principales desventajas de un trie respecto a una tabla hash son:

- En ciertos casos pueden ser más lentos que las tablas hash en la búsqueda de datos, especialmente si los datos son consultados desde dispositivos de almacenamiento secundario, como disco duro, donde el tiempo de acceso es elevado con respecto a memoria principal.
- A menudo los tries son más ineficientes respecto al espacio que las tablas hash.
- Los tries no suelen estar disponibles con las herramientas de desarrollo software, todo lo contrario que las tablas hash.
- No es sencillo representar todas las claves como cadenas, como los números reales, que pueden tener distintas representaciones en forma de cadena para un mismo número, por ejemplo 1, 1.00, 1.000, +1.000, etc.

3 Estado del Arte

La música, al contrario que otras formas de expresión artística como el dibujo o la pintura, es un medio que se presta a estudio al estar formada por una serie de elementos que pueden formalizarse y ser representados en un ordenador. Desde los comienzos de la IA, han existido trabajos que trataban de aplicar los últimos avances en algoritmos y estructuras de datos a la música. Estos trabajos pueden agruparse en tres vertientes diferentes:

- Composición: Consistente en la generación automática de melodías y armonías.
- Improvisación: Referente a la generación automática de variaciones de una base musical dada.
- Interpretación: Concerniente a la introducción de expresividad en una obra musical.

El trabajo de este PFC queda enmarcado en los sistemas de composición automática. Encontramos aquí dos grandes enfoques para abordar el problema de la composición automática: el enfoque algorítmico basado en la especificación de reglas capaces de producir composiciones musicales y un segundo enfoque, basado en la extracción de patrones, que trata de modelar ejemplos.

3.1 Sistemas de composición musical

Los primeros a la hora de crear el primer trabajo completo de música asistida por ordenador usando algoritmos de composición fueron Hiller e Isaacson [Hil]. Su trabajo de 1958 “Illiac Suite”, fue creado a partir de notas generadas de manera pseudo-aleatoria utilizando cadenas de Markov y reglas heurísticas basadas en armonía clásica y contrapunto. Se verificaba que las notas elegidas cumplieran las restricciones, en cuyo caso se añadían a la melodía. En caso de que la nota elegida no cumpliera las reglas y no se pudiese encontrar ninguna otra que las satisficiera se empleaba un mecanismo de backtracking que eliminaba toda la composición hasta ese punto para comenzar un nuevo ciclo de ejecución. A partir de esta obra surgieron nuevas composiciones basadas en cadenas de Markov con un resultado discreto, debido en gran parte a las propias limitaciones de esta técnica.

Sin embargo, no todos los sistemas de composición se han basado en métodos probabilísticos. Un buen ejemplo es el trabajo de Moorer [Moo] en 1972, basado en la generación de melodía tonal. El programa de Moorer generaba melodías simples, utilizando progresiones armónicas y repetición de patrones de notas. Este enfoque trata de emular la composición humana mediante técnicas heurísticas de un modo más cercano que las cadenas de Markov.

Siguiendo la línea de evitar el uso de probabilidades a la hora de componer, el trabajo de Levitt [Lev] en 1993 evita emplearlas, ya que para Levitt el uso de procedimientos

aleatorios dificulta más que ayuda en la tarea de componer y representar estructuras musicales simples. Por ello su trabajo se enfoca en la generación de estructuras para diferentes estilos musicales, basándose en relaciones de restricciones que denominaba “plantillas de estilo”.

El pionero en la aplicación de técnicas de IA para la composición musical fue Rader [Rad] en 1974. Este propone un sistema que se basa en reglas para decidir que notas y acordes pueden ubicarse juntos para formar melodías con cierta coherencia musical. En este sistema existen una serie de reglas de aplicabilidad con un peso determinado que según en que condiciones deciden que regla generativa se emplea. Esto constituye el primer ejemplo de uso de metaconocimiento en el contexto de los sistemas de composición automáticos.

Cabe destacar que otros nombres destacados dentro de la Inteligencia Artificial como Herbert Simon y Marvin Minsky también contribuyeron dentro del campo de la composición con ordenadores. Reseñable es el trabajo de Simon y Sumner [Sim] en 1968, los cuales mediante un método de inducción de patrones lograron descubrir patrones musicales que se hayan implícitos dentro de algunas composiciones. Este trabajo es uno de los precursores dentro de la modelización de la música. Minsky [Min] por su parte realizó escritos en donde planteaba nuevos enfoques, orientados a la creación de múltiples agentes cada uno especializado en una cualidad musical, los cuales podrían reconocer largas secciones musicales.

En 1969 Rothgeb [Rot] desarrolla un programa en Snobol enfocado a resolver el problema de la armonización de las líneas de bajo (dada una línea de bajo, inferir los acordes y la voz que la acompañan). Para ello utiliza un conjunto de reglas tales como “si el bajo de una tríada desciende un semitono, entonces la siguiente nota de bajo tiene una sexta”. El objetivo principal de Rothgeb no era desarrollar la composición automática en sí, sino probar la sonoridad computacional de dos teorías de armonización de bajos del siglo XVIII.

Una de los trabajos más completos sobre armonización es la de Ebcioglu [Ebc] en 1993. Este desarrolla un sistema experto, CHORAL, que armoniza corales según el estilo de J.S. Bach. El sistema a partir de una melodía dada produce la armonización correspondiente a partir de reglas heurísticas y una serie de restricciones. El sistema fue implementado utilizando un lenguaje de programación lógica creado por el propio Ebcioglu. Un aspecto importante de este trabajo es el uso de primitivas lógicas para representar los distintos puntos de vista de la música (acordes, vista melódica, etc.). Este conjunto de primitivas lógicas permite representar grandes cantidades de conocimientos musicales complejos.

MUSAT, creado por Bharucha [Bha], usa redes neuronales para aprender modelos armónicos y fue diseñado para capturar cualidades musicales de intuiciones armónicas. Por ejemplo, una de las cualidades de un acorde dominante es crear la expectativa de la tónica en quien lo escucha. Los compositores pueden decidir satisface o no esta expectativa según su criterio. MUSAT es capaz de aprender estas cualidades y generar diferentes grados de expectativas en un contexto armónico dado.

En 1993 Feulner [Feu] crea HARMONET, un sistema que afronta el problema de la armonización utilizando una combinación de técnicas de satisfacción de restricciones y redes neuronales. Las redes neuronales aprenden lo que se conoce como funciones armónicas y las restricciones se usan para completar ciertas partes de los acordes. Este trabajo sería ampliado por el sistema MELONET (Hornel y Dagenhardt 1997, Hornel y Menzel 1998). MELONET usa una red neuronal para aprender y reproducir estructuras de alto nivel en secuencias melódicas.

Pachet y Roy [Par] también usaron técnicas de satisfacción de restricciones para armonizar. Para ello se basan en que en que el conocimiento de la melodía y la armonización imponen restricciones sobre los posibles acordes. Sin embargo, la eficiencia del resultado es un problema en estos sistemas basados únicamente en satisfacción de restricciones.

El trabajo de Sabater, Arcos y López [Sal] en 1998 enfoca el problema de la armonización combinando el uso de reglas y CBR. Este enfoque se basa en una máxima, esta es que “las reglas no hacen la música, sino que es la música la que hace las reglas”, por lo que los sistemas basados exclusivamente en reglas suelen fallar. Por ello este sistema intenta armonizar primero una melodía buscando casos similares ya armonizados, sino encuentra ninguno aplica las reglas de armonización. En caso de no encontrar ninguna regla para aplicar, el sistema falla y realiza backtracking hasta el punto de decisión anterior. Una ventaja de este sistema es que las piezas armonizadas pueden ser usadas como ejemplos para posteriores ejecuciones.

MUSE, desarrollado por Schwanauer [Sch], es un sistema de aprendizaje que extiende una pequeña serie de restricciones sobre la voz principal, aprendiendo un conjunto de reglas sobre voz principal y secundaria. Este sistema aprende reordenando la agenda de reglas y su particionamiento para satisfacer un conjunto de restricciones sobre la voz principal. MUSE fue capaz de aprender satisfactoriamente algunas de las reglas estándar básicas sobre voces principales incluidos en libros de tonalidad musical tradicionales.

Morales y Manzanares [Mms] desarrollaron en 2001 un sistema llamada SICIB capaz de componer música a partir de movimientos corporales. Este sistema usa sensores de datos colocados en el bailarín y reglas Prolog para acoplar los gestos con la música. La arquitectura de este sistema permite interpretaciones en tiempo real.

No obstante, el trabajo más conocido utilizando la composición por ordenador y la IA es el realizado por David Cope [Cop] en 1987, el EMI. Este trabajo se centra en crear obras según el estilo de un determinado compositor y para ello emplea partituras de un compositor (al menos 2) en las que se buscan patrones recurrentes. Mediante estos patrones descubiertos, y usando una partitura original del compositor para fijar la localización de los patrones, el sistema crea nuevas obras basadas en el estilo del compositor. El resultado obtenido por este sistema se ajusta con bastante similitud al estilo del autor que intenta emular.

Recientemente ha aparecido un trabajo muy relacionado al objeto de estudio del proyecto [Exp] que ha servido como referencia. En este trabajo, a la hora de extraer estos patrones en sistemas de composición de este tipo, se usan cadenas de Markov [Hil], las cuales

representan los patrones extraídos de obras ya existentes, y en las cuales la probabilidad de pasar de un patrón a otro se calculan en función de estos mismos. En cada patrón ha de escogerse una nota inicial y duración inicial, para a partir de ellas crear la secuencia de notas y duraciones que lo represente. La elección de estos parámetros iniciales y la probabilidad de transitar a otro patrón concreto viene determinada por la información aprendida de las secuencias musicales de las cuales se extraen los patrones.

4 Planificación y presupuesto

En este apartado se va a tratar todo lo referente a la planificación seguida a la hora de realizar el proyecto, así como la metodología que se ha escogido para su desarrollo.

4.1 Planificación y metodología de desarrollo

La realización del proyecto ha seguido un desarrollo iterativo e incremental, de modo que en cada fase del proyecto hubiese una versión anterior del software que sirviera de base y experiencia para posteriores desarrollos. En cada etapa del proyecto se ha procurado tener una versión ejecutable y usable de la aplicación, de modo que se pudiera ver el estado real del sistema.

Durante el desarrollo, se establecieron una serie de reuniones periódicas con los tutores cada 15 días o 7 días, en función de lo que el proyecto requería en ese momento. Para cada una de estas reuniones se establecían una serie de objetivos a cumplir, de modo que el seguimiento sobre el desarrollo fuese continuo y cercano.

El proyecto comenzó a principios de Abril de 2010. La etapa de desarrollo del software finalizó a finales de Abril de 2011 y la documentación se realizó entre Mayo y Noviembre de 2011. El tiempo dedicado a la realización del proyecto, excluyendo periodos sin actividad, ha sido de 13 meses. Se adjunta el diagrama de Gantt relativo al desarrollo del proyecto:

Diagrama de Gantt

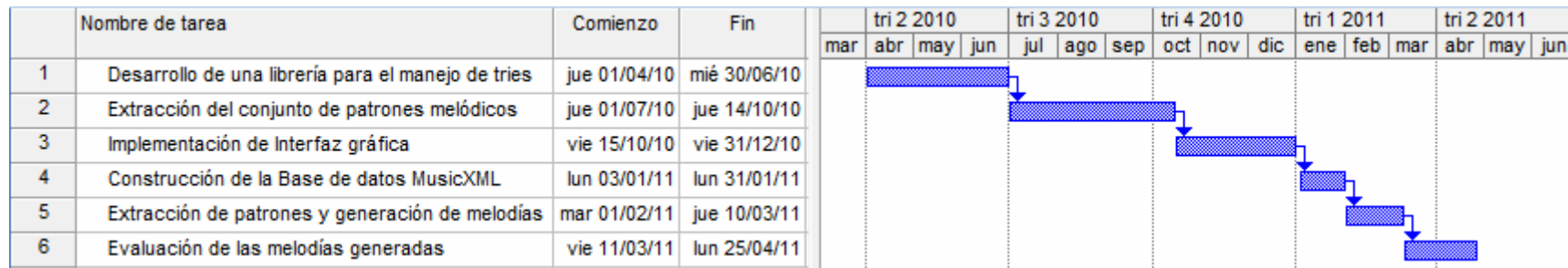


Figura 15 Diagrama Gantt de los módulos desarrollados

Tal y como se ve en la figura 15, las tareas desarrolladas no han sido realizadas en paralelo, ya que para empezar el desarrollo de un nuevo módulo era necesario acabar con la implementación del módulo anterior. Por último, se aprecia que el tiempo de desarrollo del módulo Interfaz ha sido el más breve, ocupando los módulos Trie y Extraer la mayor parte de este tiempo debido a que en ellos reside el peso y la mayor parte del proyecto.

Estimación del coste del proyecto con COCOMO

Para estimar el coste que tiene este proyecto vamos a utilizar la metodología COCOMO. Este método de estimación de costes software incluye tres submodelos, ofreciendo cada uno de ellos un nivel de detalle y aproximación, cada vez mayor, a medida que avanza el proceso de desarrollo de software: básico, intermedio y detallado.

A la vez, cada submodelo anteriormente mencionado se divide en modos que representan el tipo de proyecto, pudiendo ser:

- **modo orgánico**: El tamaño del software varía desde unos pocos miles de líneas (tamaño pequeño) a unas decenas de miles (medio). El equipo de desarrollo es reducido y el entorno estable y relajado.
- **modo semilibre o semiencajado**: corresponde a un esquema intermedio entre el orgánico y el rígido; el grupo de desarrollo puede incluir una mezcla de personas experimentadas y no experimentadas.
- **modo rígido o empotrado**: el proyecto tiene fuertes restricciones, que pueden estar relacionadas con la funcionalidad y/o pueden ser técnicas. El problema a resolver es único y es difícil basarse en la experiencia, puesto que puede no haberla.

Una vez conocidas las características de este modelo, vamos a tratar de ubicar el presente proyecto para realizar la estimación de coste.

Se ha decidido que el submodelo más adecuado para el proyecto es el de orgánico, combinado con el modelo de aplicación básico, tanto por su tamaño (alrededor de 5100 líneas de código) como por el entorno en el que se ha desarrollado, siendo este estable y sin presión por parte de los usuarios.

Para realizar los cálculos necesarios me voy a basar en la siguiente tabla que proporciona COCOMO para modelos básicos:

Tipo de Proyecto	a	b	c	d
Orgánico	2.4	1.05	2.5	0.38
Semilibre	3.0	1.12	2.5	0.35
Empotrado	3.6	1.2	2.5	0.32

Tabla 3 Tabla de valores COCOMO

Y las siguientes fórmulas:

- El esfuerzo aplicado en persona-mes: $E = a \cdot KLOC^b$ siendo KLOC el número de líneas de código.
- El tiempo de desarrollo en meses: $D = c \cdot E^d$
- El número de personas necesarias: $P = E/D$

Una vez conocidos los parámetros a utilizar, el resultado es el siguiente:

- Esfuerzo aplicado en persona-mes: $E = 2,4 \cdot (5,1)^{1,05} = 13,27$ meses/hombre
- Tiempo de desarrollo en meses: $D = 2,5 \cdot (13,27)^{0,38} = 6,67$ meses.

Obtenidos los resultados del esfuerzo y el tiempo de desarrollo, el número total de recursos necesarios para el desarrollo es:

$$P = 27,64 / 8,82 = 3,13 = 1,989 \approx 2 \text{ personas.}$$

4.2 Presupuesto

Para la realización de este presupuesto se ha tenido en cuenta los diversos recursos que han sido necesarios para el desarrollo del proyecto. El coste total del proyecto asciende a **18.827,58 €**. A continuación se detallan todos los gastos incluidos en el presupuesto.

Gastos de personal

Basándome en la previsión realizada con COCOMO, para la realización del proyecto en el tiempo establecido son necesarias 2 personas. Estas 2 personas tendrán cada uno un rol específico que se muestra a continuación:

- Analista: Se encargará de identificar los requisitos necesarios para el desarrollo del sistema así como de analizar el problema propuesto y diseñar una solución. También se encargará de establecer las pruebas que evaluarán el funcionamiento de la aplicación. Por último ejercerá la labor de jefe de equipo elaborando toda la documentación referente al proyecto, además de coordinar la labor del programador junior.
- Programador Junior: Se encargará de realizar las tareas de programación que establezca el jefe de equipo así como de la realización de las pruebas.

En función de estas necesidades, el gasto derivado de la contratación del personal es el siguiente:

Categoría profesional	Nº Recursos	Salario base	Meses	Total
Analista	1	1.569,25€	7	10.984,75€
Programador Junior	1	987,69€	7	6.913,83€
Total				17.898,58€

Tabla 4 Gastos contratación de personal

Cabe decir que el salario que deben percibir los 2 contratados se ajusta a lo establecido en el XVI Convenio colectivo estatal de empresas de consultoría y estudios de mercados y de la opinión pública, anexo III [BOE].

Equipos Informáticos

Para la realización de este proyecto ha sido necesaria la adquisición de un ordenador portátil Dell Inspiron 1750 cuyo coste ha sido de 599€. El resto de equipo informático utilizado tiene una antigüedad suficiente para considerar que están amortizados por lo que no supone coste alguno.

Licencias

Para la obtención de las partituras de MusicXML se ha utilizado el programa Guitar Pro 6, cuyo coste de licencia ha sido de 100 €.

El resto de software utilizado para la realización del proyecto es software libre, por lo que no suponen coste adicional.

Material Fungible

Se establecen los siguientes gastos en materiales fungibles.

Concepto	Precio
Papel	25 €
Material de escritura	15 €
Dispositivo de almacenamiento externo	40 €
Imprenta y encuadernación	120 €
CDs y DVDs	30 €
Total	230 €

Tabla 5 Gastos material fungible

5 Trabajo realizado

En el presente apartado se va a detallar todo el trabajo realizado para completar el desarrollo de la aplicación. Dentro de él se trata el flujo que sigue el sistema a la hora de crear una melodía, así como la arquitectura del sistema, los paquetes que lo componen y las relaciones existentes entre ellos.

5.1 Flujo del sistema

A partir del diagrama Gantt presentado en la figura 15 se va a describir el flujo de datos del sistema, de modo que se pueda apreciar los pasos dados a la hora de generar las melodías.

Fase Extracción de Información

El sistema inicia su ejecución extrayendo toda la información de las partituras de MusicXML del estilo que se ha elegido. Para ello se tratan todas las partituras, una a una, obteniendo frases formadas por intervalos de altura o duración. El criterio tenido en cuenta para delimitar una frase ha sido o bien entender como frase las notas que hay entre dos silencios o que la frase tuviera una duración máxima de 2 compases. Veamos un ejemplo para aclarar el trato que hemos dado a los intervalos:



Figura 16 Ejemplo de Intervalo de duración [Mus]

En este caso tenemos un intervalo de duración, en el que la duración de la primera figura es una redonda, con valor de 4 negras, y la duración de la figura 2 es una blanca, con valor de 2 negras. De esta manera el intervalo de duración entre estas 2 notas es de $2/4=1/2$, es decir, la variación de la duración entre las dos figuras. A cada posible variación se le ha asignado una letra, de manera que una frase de duración extraída de una partitura como esta:



Figura 17 Ejemplo de frase de intervalos de duración

Se vea representada del siguiente modo: **IBHSAHAAA**

Siendo cada letra la representación de un intervalo de duración y constituyendo todas juntas una frase.

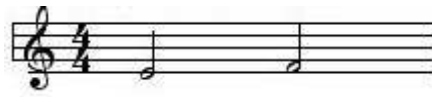


Figura 18 Ejemplo de Intervalo de Altura [Mus]

En la figura 18 podemos observar un intervalo de altura, donde la primera nota es Mi y la segunda Fa. Entre estas dos notas existe un intervalo de altura de 2ª menor ascendente, al haber una diferencia de un semitono entre ambas notas. A cada posible intervalo de altura se le asigna una letra al igual que se hizo con los intervalos de duración, pero en este caso se ha de diferenciar entre intervalo ascendente y descendente. Un intervalo descendente se refiere a cuando se pasa de una nota de mayor altura a una nota de altura menor, siendo el intervalo ascendente el caso opuesto. Para diferenciarlos, las frases de altura contienen letras en mayúscula y minúscula, correspondiendo las primeras a intervalos ascendentes y las segundas a los descendentes. Debido a ello las frases de altura obtenidas de partituras como esta:



Figura 19 Ejemplo de frase de intervalos de altura

Tienen el siguiente aspecto: **HacDCaa**

Representando cada letra un intervalo y formando todas juntas una frase.

La información sobre las distintas frases de intervalos de duración y de altura extraídas de los ficheros Music XML se almacena en 2 estructuras de datos trie. Se generan 2 tries, uno para las frases con los intervalos de altura y otro para las frases con los intervalos de duración. En estas estructuras se van insertando las frases, de modo que se generen 2 tries en las que en sus nodos hojas se puede ver el número de veces que aparece cada frase.

El proceso para construir los tries es el siguiente: las frases (sucesión de intervalos de altura o duración) se irán insertando una a una en el trie, comprobando antes si la frase a insertar ya existe en el trie, en cuyo caso se incrementará el número de apariciones. Una vez acabada esta operación tendremos un trie que contiene todas las frases (en este caso serán nodos hojas del trie), generadas en la extracción de información las partituras, ordenadas y con el número de aparición de cada una asociado. Veamos unos ejemplos:

Trie de altura

```
[B]
BBebcDEFDbB 10
[C]
BCCaBbcc 20
BCaCcCb 20
BCddBCdB 10
[b]
BbBbBbB 10
[b]
[B]
[F]
[D]
[b]
[c]
[C]
[c]
[b]
[B]
BbbBFDbcCcbBCF 80
BbbBFDbcCcbBDEebBbbBDDdbBbbBDF 20
BbbBFDdbBbbBDFfcCcbBDEebBbbBED 20
BbbBHcbBCcbB 20
BbbBIbbbbbbbbbbCcbBMbbbbbbbbbb 20
BhcbccCCBCCCB 10
```

Figura 20 Salida de un trie de intervalos de altura obtenido de la obra Preludio 6

En este ejemplo tenemos todas las frases que comienzan por la letra “B” de un trie de intervalos de altura, ordenadas según el camino que recorren hasta alcanzar el nodo hoja. Este nodo hoja o frase tiene asociado el número de apariciones en el total del trie.

Como se puede ver en el trie, tenemos 11 tipos de frases distintas con distintos números de apariciones, como la frase BCCaBbcc la cual aparece en la obra 20 veces. Este número de apariciones se va a utilizar posteriormente a la hora de generar las melodías, ya que en función de ello se puede ver lo habitual que es una determinada frase en un género concreto y la probabilidad que tendrá de salir en la generación de la melodía.

Trie de duración

```
[I]
[A]
[A]
[A]
IAAA1 3
[A]
[A]
[A]
[A]
[A]
[A]
[A]
[A]
IAAAAAA1 2
IAAAAAA1 2
IAAAAAADAAA1 1
IAAADIAAA1 4
```

Figura 21 Salida de un trie de intervalos de duración obtenido de la obra *Only for the weak*

En la figura 21 tenemos todas las frases que empiezan por I en un trie de altura, ordenadas según el camino que recorren hasta llegar al nodo hoja, al igual que con el trie de altura. Se puede observar que todas las frases contienen letras en mayúsculas, ya que no se distingue entre intervalos ascendentes y descendentes. Es destacable que todas las frases de duración terminan con el número “1”, símbolo de parada elegido expresamente para delimitar el fin de este tipo de frases. La elección de este símbolo viene motivada por la posibilidad de que una frase de intervalos de altura acabe al llegar a un silencio o a que llegue al límite de duración de dos compases. Para contemplar estos posibles casos, a las frases de intervalos de altura se les añade la letra “n” al final de ellas durante el proceso de generación de nuevas frases. Este símbolo podrá ser un silencio o no dependiendo de una función aleatoria que lo determina a la hora de generar nuevas progresiones de notas. Para equiparar la longitud de las frases de intervalos de duración con las de altura, obtenidas simultáneamente, se les ha añadido el símbolo de parada “1”. Por último, en cada nodo hoja queda reflejado el número de apariciones.

Como se puede ver en el trie, este contiene 5 frases distintas con un número de apariciones determinado, como la frase IAAA1 la cual aparece 3 veces en esta obra. Este número de apariciones se va a utilizar posteriormente a la hora de generar las melodías, ya que en función de ello se puede ver lo habitual que es una determinada frase en un género concreto y la probabilidad que tendrá de salir en la generación de la melodía.

Fase Creación de Patrones

Una vez realizado la extracción de información de las partituras y con la información obtenida en los tries, se generan nuevas frases a partir de las existentes. La concatenación

de 2 o más de estas frases forman los patrones, aunque puede darse el caso de que un patrón este formado por una única frase.

Para generar un patrón de una melodía, lo primero que se hace es elegir una nota aleatoria, que será la nota de inicio de la melodía. Hecho esto, se elige una frase de intervalos de altura de forma aleatoria de entre todas las obtenidas de las partituras y almacenadas en el trie de intervalos de altura. Las probabilidades de elegir una frase de intervalos de altura en particular son proporcionales a su número de apariciones registrado en el trie.

A partir de la frase de intervalos de altura elegida, se escoge una frase de duración de forma aleatoria de entre aquellas que tengan el mismo tamaño que la frase de altura. Esta frase determinará la duración de las notas que generadas anteriormente. De nuevo, las probabilidades de elegir una frase de intervalos de duración en particular son proporcionales a su número de apariciones registrado en el trie. De nuevo, generamos la duración de la primera nota de forma aleatoria al igual que con la altura, y a partir de ella construimos la secuencia de duraciones, siempre y cuando esta progresión entre dentro de los límites de duración existentes. Realizados estos pasos ya tenemos una nueva frase generada a partir de la información obtenida de las partituras. La figura 22 ilustra el algoritmo descrito para generación de nuevas frases.

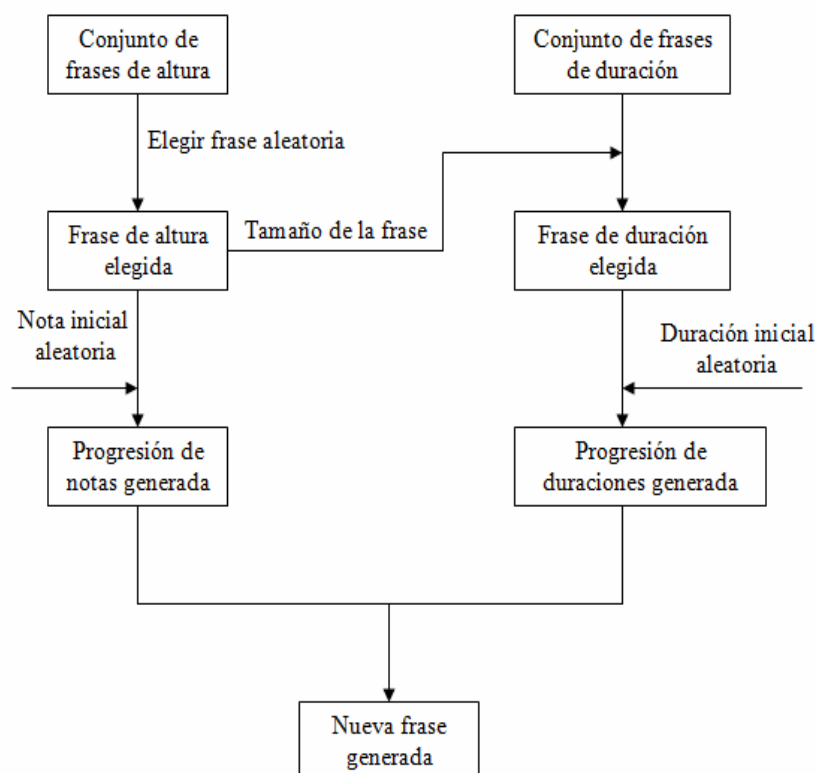


Figura 22 Algoritmo de generación de frases

Para crear un patrón, será necesario concatenar estas frases generadas hasta alcanzar la duración deseada, pudiendo darse la posibilidad de que un patrón este formado por una sola frase si esta tiene igual duración que el patrón.

El proceso para generar los patrones es el siguiente:

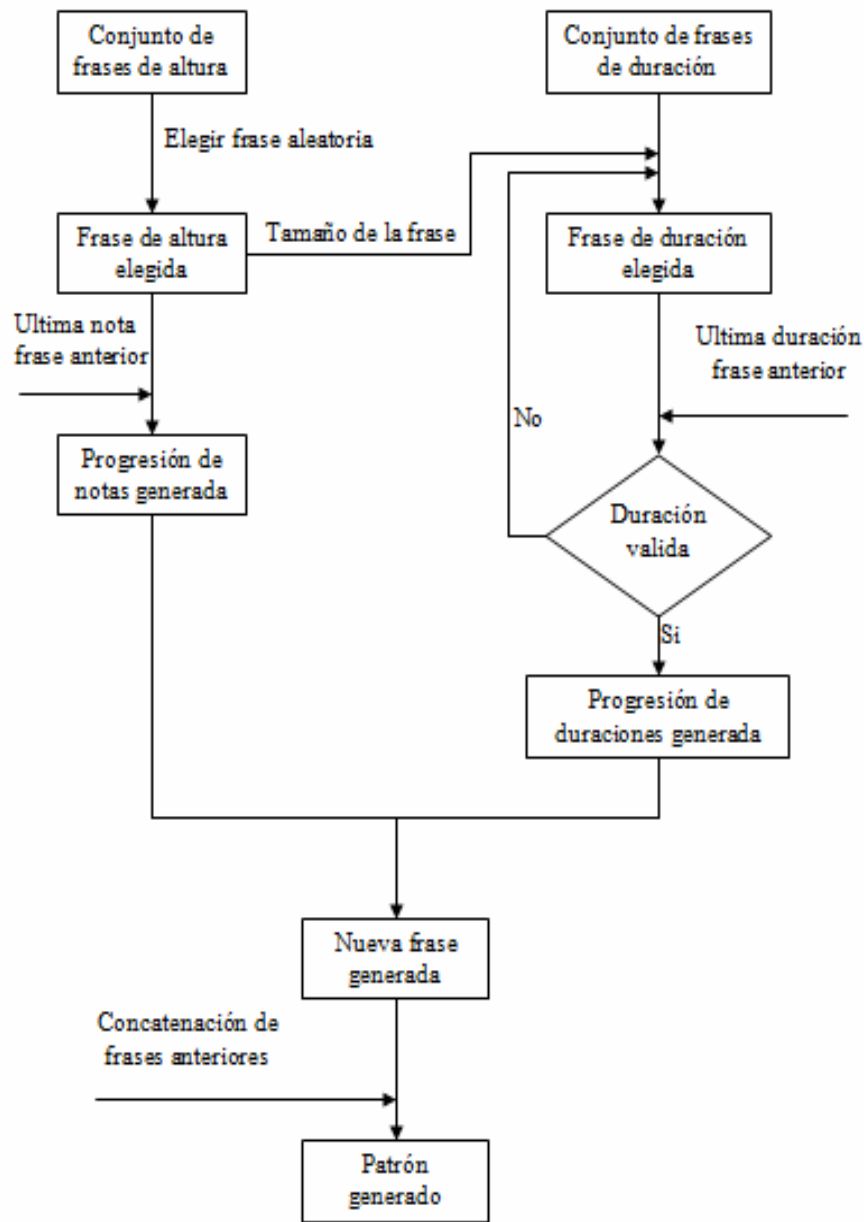


Figura 23 Algoritmo de generación de un patrón

Partiendo de una frase generada mediante el proceso representado en la figura 23, se produce la concatenación de patrones que forman un patrón. Como base para generar las siguientes frases, se toma como nota y duración de inicio las últimas de la frase anterior. Es decir, si la frase anterior su última nota es un Mi de duración una negra, la siguiente frase toma estos valores para calcular cuales son su primera nota y figura de duración en función del primer intervalo de sus frases.

Una vez elegida la frase de altura de forma aleatoria de entre todas las disponibles, se elige una frase de duración en función del tamaño de la frase de altura. Sin embargo, es posible que la frase de duración escogida no sea válida por la progresión de sus intervalos. Pongamos un ejemplo: En el proceso de generar una nueva frase, la frase de duración contiene un primer intervalo que indica que la siguiente figura tiene que doblar en duración a la anterior. En este caso, tenemos que la frase anterior acababa en una redonda, por lo que la siguiente figura tendría que durar 8 negras. Representar esto con una figura no es posible, así que esta frase no sería válida en este contexto. En estos casos donde la progresión de las duraciones no es válida, se descarta la frase elegida y se vuelve a elegir una entre aquellas con el mismo tamaño que la frase de altura hasta encontrar una adecuada. Al obtener una progresión de intervalos de altura y una progresión de intervalos de duración válida, estas se combinan para formar una nueva frase. Esta nueva frase se concatena con las frases previamente generadas, en caso de haberlas, para formar un nuevo patrón. Este proceso se repite hasta alcanzar la duración definida.

Este proceso se repite cada vez que se genera un nuevo patrón, los cuales a su vez constituyen las melodías MIDI que se obtienen como salida de la aplicación.

Fase Generación de Melodía

Llegados a este punto del flujo del sistema, solo queda combinar los patrones creados para obtener la melodía MIDI generada. A través del interfaz, el usuario ha podido escoger el estilo, duración de los patrones y la secuencia de estos mismos que forman la melodía final. Cada patrón se identifica con una letra acompañado de un número indicando su duración. Por ejemplo, si tenemos el patrón A24, la letra A lo identifica y el número representa que su duración es de 24 negras. En función de los patrones generados, el usuario escoge una secuencia de patrones que formen el MIDI final. Dados por ejemplo los patrones A24, B16 y C12, una posible secuencia de patrones puede ser:

A24+B16+A24+C12+B16+C12

En donde la melodía se compone de los patrones concatenados según el orden que se ve, primero el patrón A24, a continuación el B16, luego el A24 y así hasta terminar con el patrón C12. Una vez decidido el orden de los patrones, se junta esta secuencia en un mismo fichero de texto, de modo que formen una sola melodía y genere el acompañamiento que le corresponde a este estilo musical. Obtenido el fichero que contiene la melodía y su acompañamiento, el último paso es generar la melodía MIDI correspondiente a ese fichero de texto. Cabe decir que cada patrón creado puede ser reproducido, de manera que el escucharlo ayude en su posible inclusión en la melodía final o en su eliminación, según las preferencias del usuario.

5.2 Arquitectura del sistema

En este apartado vamos a ver la arquitectura empleada en el sistema, detallando todos los paquetes y clases necesarios para realizar todo el proceso desde la extracción de información de las partituras en MusicXML hasta la generación de la melodía MIDI. Veamos la relación entre los principales paquetes que componen la aplicación:

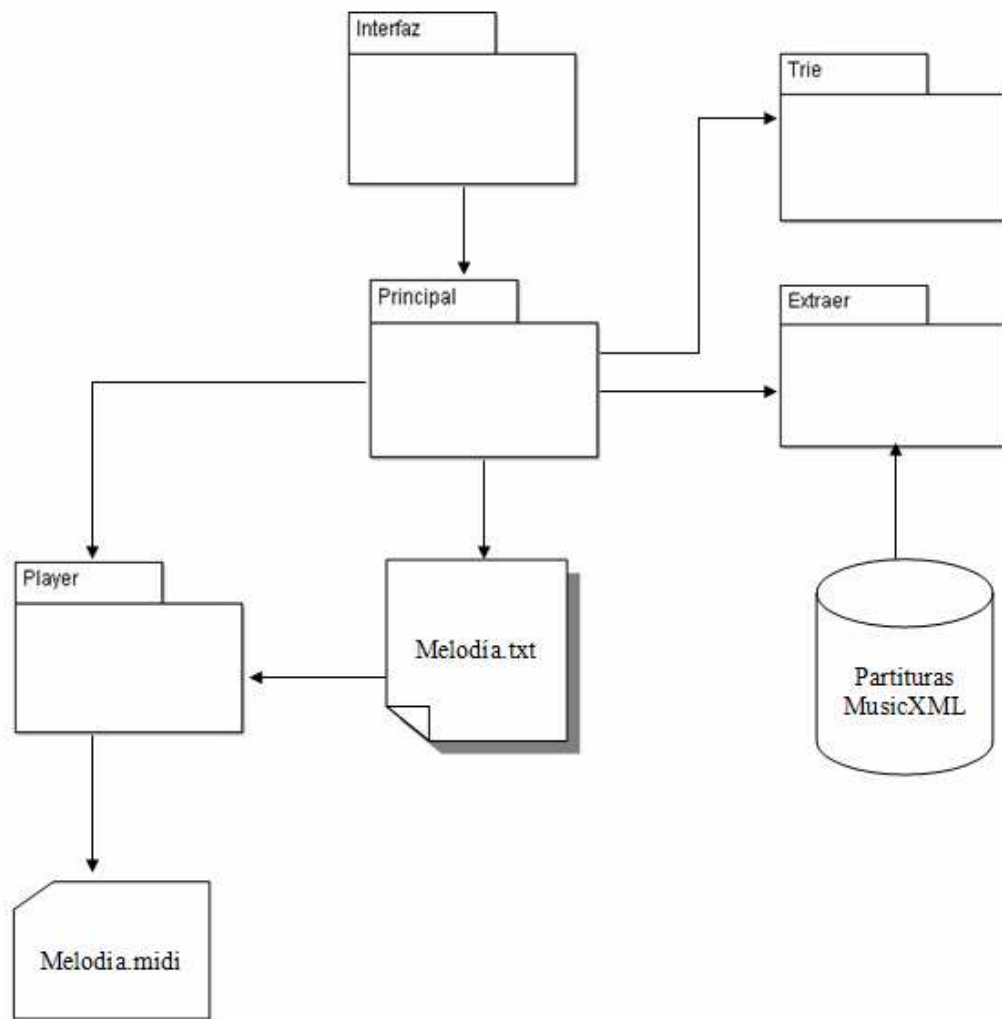


Figura 24 Relación de paquetes que componen el sistema

Como se ve en la figura 24, el paquete Principal es la parte central de la aplicación, comunicándose y haciendo llamadas a los distintos paquetes. Este paquete a su vez tiene como entrada las instrucciones que recibe desde el paquete Interfaz, tales como el estilo musical a modelar o la secuencia de patrones que compondrá el MIDI final. Como salida, genera un archivo .txt que contiene la melodía creada. El paquete Principal hace una llamada al paquete Player para generar el MIDI, recibiendo como entrada el archivo .txt generado durante todo el proceso de ejecución y produciendo como salida el archivo MIDI correspondiente.

Respecto al paquete Extraer, tiene como entradas las llamadas del paquete Principal que determinan duración y composición de patrones a generar y además la partitura con la que se trabaja. Una vez extraída la información necesaria de las partituras, el paquete Principal se comunica con el paquete Trie, recibiendo estos datos como entrada.

5.2.1 Paquete Principal

Es el paquete principal de la aplicación, desde el que se leen los valores de entrada del programa, tales como el estilo musical deseado, y desde el que se hacen llamadas a todos los paquetes necesarios para la creación de la melodía MIDI del estilo escogido.

Su diagrama de clases es el siguiente:

Main
<pre> public static void BorrarFichero(File){} public static String DameInstrumento(String){} public static String CreaPatron(String,String,String,String,String,String,double,int,String,int Cancion){} public static String UnionPatrones(Patrones){} public static int AjustarOctavas(String){} public static String AjustaPatron(int,String){} public static void Reproducir(String){} </pre>

Figura 25 Clase Main

Como se puede observar, el paquete Principal esta formado únicamente por la clase Main, el cual contiene los siguientes atributos y métodos:

Atributos:

- Esta clase no tiene atributos propios.

Métodos:

- `borrarFichero()`: método que elimina un fichero una vez dado.
- `DameInstrumento()`: método que nos devuelve el instrumento que interpretará la melodía según el estilo elegido.
- `CreaPatron()`: método principal de la clase. A partir de los datos parseados y el tratamiento de estos en el trie, genera un patrón del estilo deseado.
- `UnionPatrones()`: método que une los patrones generados, según el orden elegido por el usuario.
- `AjustarOctavas()`: método que ajusta el valor de la octava inicial en la que empieza un patrón. Esta función es complementaria de `AjustaPatrones()`.
- `AjustaPatrones()`: método que ajusta el valor de la octava inicial en la que empieza un patrón.
- `Reproducir()`: método que invoca al Player para crear y reproducir MIDI.

5.2.2 Paquete Extraer

En este paquete están todas las estructuras y todas las clases necesarias para extraer la información de las partituras en MusicXML. Esta compuesto por una serie de clases, que se comunican tal y como se muestra en el diagrama de clases.

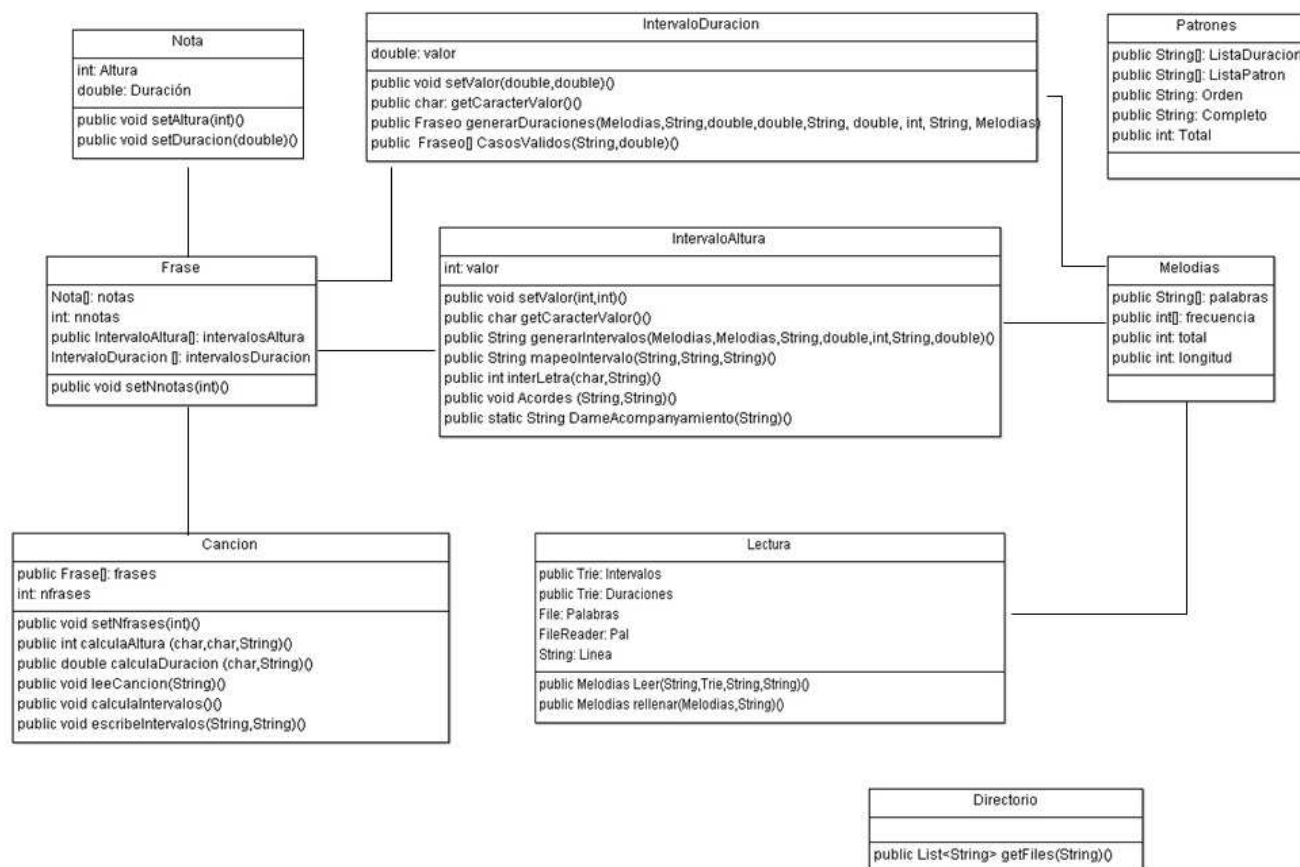


Figura 26 Diagrama de clases módulo Extraer

- La clase **Nota** es la representación formal de una nota musical, por lo que sus atributos comprenden la Altura y la Duración. Sus dos métodos, setAltura() y setDuración() son utilizadas para dar valor a los atributos anteriormente mencionados.

Nota
int: Altura double: Duración
public void setAltura(int){} public void setDuracion(double){}

Figura 27 Clase Nota

- La clase **Frase** es la representación de lo que formalmente se entiende por frase en un contexto musical. En este caso, se ha decidido que una frase sea las notas comprendidas entre 2 silencios. En caso de no haber silencios, se delimitará la frase a la duración de 2 compases. Como se puede ver, esta clase es un array de notas, almacenando también el número de estas que contiene. Los atributos intervaloAltura e intervaloDuracion guardan los intervalos de altura y duración obtenidos de esta frase.

Respecto a los métodos de esta clase, setNnotas da valor al atributo que marca el número de notas de la frase.

Frase
Nota[]: notas int: nnotas public IntervaloAltura[]: intervalosAltura IntervaloDuracion []: intervalosDuracion public void setNnotas(int){}

Figura 28 Clase Frase

- La clase **Canción** es la representación de lo que normalmente se entiende por una canción. Está formada por un array de frases, las cuales en conjunto forman una canción. Respecto a sus métodos, a continuación las vemos con más detalle:
 - setNfrases(): Este método da valor al atributo que marca el número de frases que contiene la canción.
 - calculaAltura(): Este método, dadas dos notas, nos devuelve el intervalo de altura correspondiente.
 - calculaDuracion(): Este método, dadas dos notas, nos devuelve el intervalo de duración correspondiente.
 - calculaIntervalos(): Método que calcula los intervalos de duración y altura de una frase.
 - escribeIntervalos(): Este método escribe el resultado de la transcripción de una frase en intervalos de duración y altura en los ficheros en los ficheros “Altura” y “Duración”.
 - leeCancion(): Método principal de la clase. Este método, dada una partitura en MusicXML que se le pasa desde el paquete Principal, parsea la información que contiene referente a notas, duraciones, compases, etc. y la almacena en las clases previamente nombradas.

Cancion
<pre>public Frase[]: frases int: nfrases</pre>
<pre>public void setNfrases(int){} public int calculaAltura (char,char,String){} public double calculaDuracion (char,String){} public void leeCancion(String){} public void calculaIntervalos(){ } public void escribeIntervalos(String,String){}</pre>

Figura 29 Clase Cancion

La clase **IntervaloDuración** es la representación de lo que en el proyecto se entiende por un intervalo de duración, como su nombre indica.

Respecto a sus principales métodos, procedo a explicarlas a continuación:

- setValor(): Método que da valor al atributo “valor”.
- getCaracterValor(): Método que calcula el intervalo de duración entre 2 notas.
- generarDuraciones(): Método que elige una frase, que representa una serie de intervalos de duración, y la transforma en la duración que tendrán las sucesivas notas de una frase de la melodía resultante.
- Casosvalidos(): Al generar las duraciones que tendrá una determinada frase, no siempre obtendremos una progresión valida. Pongamos por ejemplo, una frase cuya primera figura sea una redonda, y un intervalo de duración que indique que la siguiente nota debe doblar en duración a la anterior. En este caso, se nos presenta un problema, ya que no existe ninguna figura musical que represente esta duración. Este método se encarga de verificar que progresiones son validas y cuales no.

IntervaloDuracion
double: valor
<pre>public void setValor(double,double){} public char: getCaracterValor(){ } public Fraseo generarDuraciones(Melodias,String,double,double,String, double, int, String, Melodias) public Fraseo[] CasosValidos(String,double){}</pre>

Figura 30 Clase IntervaloDuracion

- La clase **IntervaloAltura** es una representación de un intervalo entre dos notas. Consta de un atributo valor que almacena el intervalo entre 2 notas.

Respecto a sus métodos, se detallan a continuación las más importantes:

- setValor(): Método que da valor al atributo “valor”.
- getCaracterValor(): Método que calcula el intervalo de altura entre 2 notas.
- generarIntervalos(): Método que elige una frase, que representa una serie de intervalos de altura, y la transforma en la progresión de notas que tendrá cada frase de la melodía. Realiza esta operación de forma iterativa hasta que se alcanza la duración delimitada para la canción.
- mapeoIntervalo(): Este método, a partir de una nota inicial que se le pasa por parámetro y una frase que representa un conjunto de intervalos de altura, genera las progresiones de una única frase. Esta función es complementaria a generarIntervalos().
- Acordes(): Todas las melodías generadas tendrán un acompañamiento en forma de acorde al inicio de cada compás. Este método va generando esos acordes en función de la nota que suene al inicio de un compás.
- DameAcompañamiento(): El instrumento que interprete el acompañamiento variará según el estilo musical elegido, con el fin de lograr una sonoridad más adecuada. Este método elige ese instrumento en función del estilo.

IntervaloAltura
int: valor
<pre> public void setValor(int,int)() public char getCaracterValor()() public String generarIntervalos(Melodias,Melodias,String,double,int,String,double)() public String mapeoIntervalo(String,String,String)() public int interLetra(char,String)() public void Acordes (String,String)() public static String DameAcompañamiento(String)() </pre>

Figura 31 Clase IntervaloAltura

- La clase **Lectura** es la encargada de introducir la información obtenida en el parseo de las partituras en las estructuras tries. Por ello, entre sus atributos cuenta con 2 tries, uno con información sobre los intervalos de altura y otro con los intervalos de duración. Una vez se introduzcan los datos en los tries, esta clase también se encarga de extraer la información resultante de los tries y almacenarlo en instancias de la clase Melodías para su posterior tratamiento en la generación de canciones.

Sus dos métodos principales se ocupan de los aspectos anteriormente mencionados:

- Leer(): Este método lee la información obtenida del parseo y la introduce en su trie correspondiente, Intervalos o Duraciones.
- Rellenar(): Este método extrae la información del trie y la almacena ordenada en una instancia de la clase Melodias.

Lectura
public Trie: Intervalos public Trie: Duraciones File: Palabras FileReader: Pal String: Linea
public Melodias Leer(String,Trie,String,String)() public Melodias rellenar(Melodias,String)()

Figura 32 Clase Lectura

- La clase **Melodías** está diseñada para guardar y tratar la información extraída de los tries. Esta compuesta por dos arrays, frases y frecuencia, los cuales son correlativos, ya que uno almacena cada frase obtenida del trie (entendiendo frase como conjunto de intervalos de duración y altura) y la frecuencia de aparición de esta frase respecto al total contenido en el trie. El atributo total indica el número total de frases que contiene el trie, mientras el atributo longitud da el número de frases distintas que aparecen. Esta clase no tiene métodos propios.

Melodias
public String[]: palabras public int[]: frecuencia public int: total public int: longitud

Figura 33 Clase Melodias

- La clase **Patrones** se ha diseñado para guardar toda la información relativa a los patrones melódicos introducidos por el usuario, los cuales serán generados por la aplicación. En esta clase se almacenará la duración de cada patrón, su número asociado dentro de la lista de patrones introducidos, el orden de aparición de los patrones que compondrán la melodía final y el total de patrones que se han generado. Esta clase no tiene métodos propios.

Patrones
public String[]: ListaDuracion public String[]: ListaPatron public String: Orden public String: Completo public int: Total

Figura 34 Clase Patrones

- La clase **Directorio** sirve como apoyo a la hora de trabajar con las carpetas que contienen las partituras de los estilos musicales. Cabe señalar que a la hora de utilizar la aplicación, se le indicará un directorio (elegido por el usuario) que contenga las partituras de un estilo concreto, procesando una por una estos archivos para extraer la información del estilo elegido.

Directorio
public List<String> getFiles(String)()

Figura 35 Clase Directorio

5.2.3 Paquete Trie

En este paquete se define la clase Trie, utilizada para procesar la información parseada de las partituras. Las razones para emplear esta estructura son varias, comenzando por la mayor eficiencia en la búsqueda de claves, la posibilidad de ordenarlas alfabéticamente y el tratamiento de las colisiones de claves.

Como se indicó anteriormente, este paquete se compone exclusivamente de la clase Trie, la cual se procede a explicar a continuación:

Trie
int: position Trie: previous TrieEntry: at Object: entries[]
public void insert(String, Object)() public void remove(String)() public Object Busqueda(String)() public void dump(int,String)()

Figura 36 Clase Trie

Respecto a sus atributos, se definen los siguientes:

- Position: Este atributo marca la posición dentro del trie.
- Previous: Puntero a la posición precedente del elemento del trie.
- At: instancia de una clase privada de trie, utilizada cuando una clave acaba en este trie. Sus atributos nos indican la clave del nodo hoja y el número de ocurrencias dentro del trie.
- Entries[]: Atributo que nos marca el contenido de este trie.

En cuanto a sus métodos, detallamos las fundamentales para el entendimiento de la clase:

- Insert(): Este método coge una frase o clave y la inserta en el trie, comprobando antes gracias a la función Búsqueda() si esta frase ha sido insertada ya en el trie. Si se da este último caso no insertará la palabra.
- Remove(): Método que dada una frase la busca y elimina del trie.
- Búsqueda(): Dada una frase, este método recorre el trie para buscarla. En caso de que la encuentre incrementará en 1 el número de ocurrencia asociado a la frase.
- Dump(): Este método nos muestra una representación gráfica del trie generado, cono todos los nodos hoja generados y el camino recorrido por los nodos rama hasta llegar a ellos.

5.2.4 Paquete Interfaz

En este apartado vamos a definir el paquete Interfaz, el cual contiene la clase Ventana, encargada de la creación y manejo del interfaz de usuario.

Este interfaz está compuesto por una serie de componentes que nos ofrece la librería Swing, tales como botones, lista, etc.

A continuación se va a explicar la funcionalidad que desempeña cada componente, así como las relaciones existentes entre ellos.

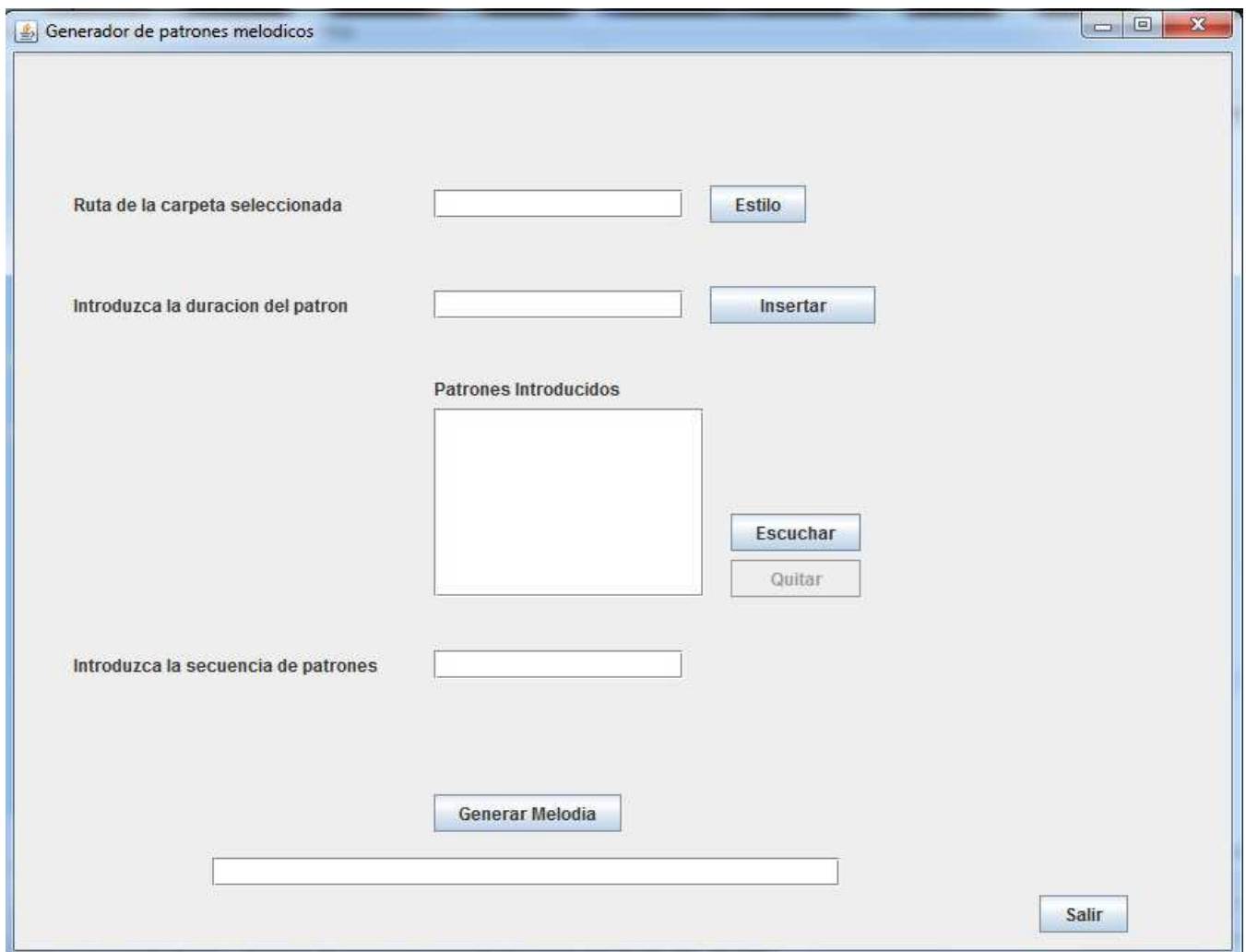


Figura 37 Paquete Interfaz

- Elección del estilo musical deseado



Ruta de la carpeta seleccionada

Figura 38 Selección de estilo

A la hora de tratar con el interfaz de usuario, lo primero con lo que se interactúa es con el botón estilo, mediante el cual elegimos el estilo musical seleccionando una carpeta previamente etiquetada que contiene partituras MusicXML del estilo a modelar.

Tal y como se ve en la imagen superior, nos encontramos con el botón estilo y un TextField que contendrá la ruta del directorio donde se encuentran las partituras del estilo musical a modelar. Al hacer clic sobre el botón estilo se dispara el evento asociado a este componente, desplegándose un FileChooser que nos permite elegir el directorio deseado.

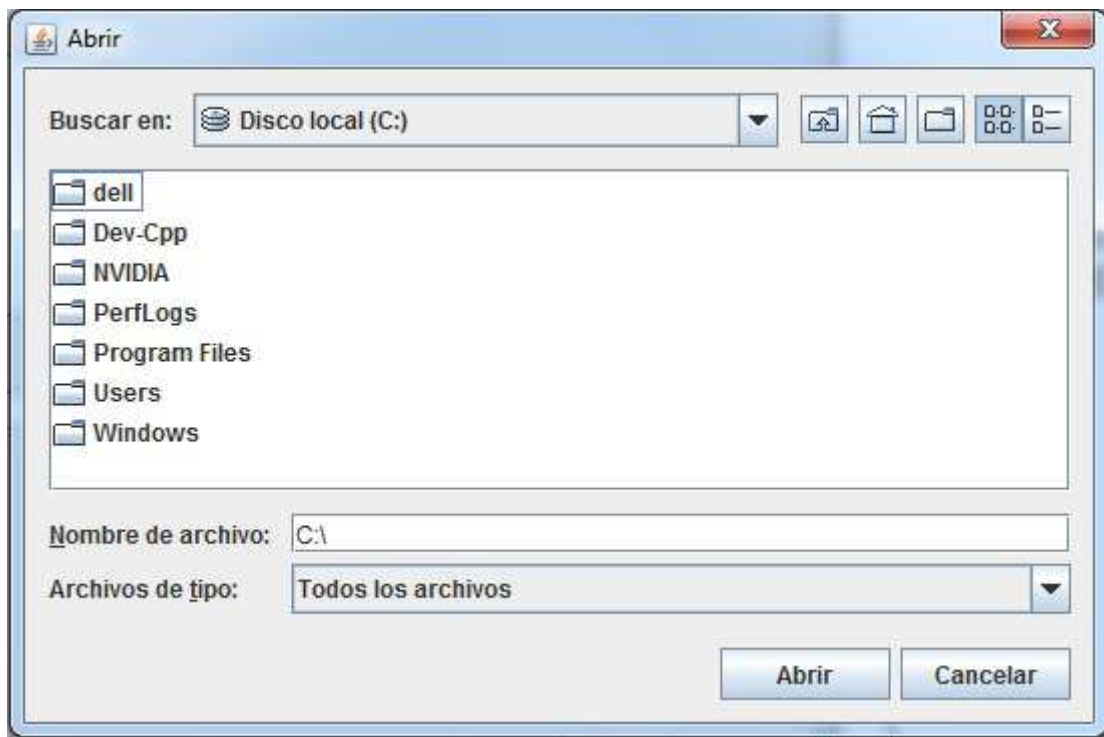


Figura 39 Elección de directorio

Una vez seleccionada la carpeta que contiene las partituras del estilo seleccionado, la ruta de esta carpeta se asocia al TextField que aparece contiguo a estilo. De este modo el usuario puede saber en todo momento que estilo musical eligió como entrada.

- Creación de patrones y estructura de la melodía:

Figura 40 Creación de patrones y estructura de la melodía

Este punto describe las estructuras y componentes necesarios para crear uno por uno los patrones que pueden formar parte de la melodía final, así como la unión de patrones que conformarán la melodía resultante.

Como se ve en la parte superior de la imagen, tenemos el botón Insertar con un TextField contiguo. En este TextField se introduce la duración en negras que tendrá el patrón creado. Al hacer clic sobre el botón Insertar se dispara el evento asociado, efectuándose una serie de operaciones que terminarán con la creación del patrón.

Primero se obtiene el estilo musical del TextField que contiene la ruta de la carpeta escogida, para a continuación llamar al método CrearPatron() perteneciente al paquete Principal. Esta función crea un patrón del estilo elegido y de la duración escogida a partir de la información obtenida de los tries. Una vez se ha creado el patrón, se inserta con su codificación correspondiente (que ayudará a distinguir patrones de misma duración) en la lista. Este proceso se puede repetir las veces que se quiera hasta tener en la lista el número de patrones que se desee.

En la parte central de la imagen tenemos la lista de patrones introducidos, con una serie de botones asociados. Estos botones presentan cada uno una funcionalidad específica que se explican a continuación:

- **Botón Escuchar:** Este botón permite escuchar la representación en MIDI del patrón creado. Cuando se hace clic sobre él, primero se comprueba si hay algún elemento de la lista seleccionado. En caso de que lo haya, coge el nombre asociado al elemento seleccionado y se lo envía al método Reproducir() perteneciente al paquete Principal, el cual llama al paquete Player, que se encargará de reproducir el patrón deseado.
- **Botón Quitar:** Este botón permite eliminar elementos de la lista. Cuando se hace clic sobre se comprueba que haya algún elemento seleccionado, eliminándolo en caso afirmativo. Cabe destacar que el botón está desactivado mientras no haya ningún elemento en la lista.

Por otro lado, el componente lista tiene otra función asociada. Si se hace doble clic sobre algún elemento de la lista se dispara otro evento, el cual introduce el elemento correspondiente al patrón en el TextField que se observa en la parte inferior de la imagen y que corresponde a la secuencia de patrones que compondrán la melodía.

- Generar la melodía:

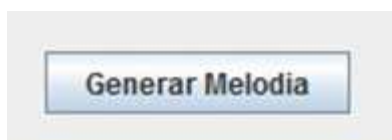


Figura 41 Generar la melodía

Por último tenemos el botón Generar Melodía y un TextField en la parte inferior. Cuando se hace clic sobre el este botón, se dispara un evento que creará la melodía MIDI. Primero se lee la información del TextField que contiene la secuencia de patrones. Con la secuencia se forma la melodía final uniendo en un solo fichero la serie de notas resultante de esta secuencia de patrones. Al acabar este proceso, se llama al método Acordes (), a través del paquete Principal. Este método generará el acompañamiento de la melodía en función del estilo que se eligió. El proceso para generar el acompañamiento es el siguiente:

- Al inicio de cada compás de 4/4 (es decir, un compás de duración 4 negras) se elegirá el acorde que sirve de acompañamiento en función de la nota que esté sonando en ese momento.
- Este acorde dura una redonda, y se elige en función de los acordes mayores que corresponden a cada nota. Cabe decir que se ha implementado una función que evita que se elija el mismo acorde de forma continuada, ya que distintas notas pueden compartir un mismo acorde mayor. De esta forma se consigue una sonoridad más variada.

Para finalizar, se hace una llamada al método Reproducir() del paquete principal pasándole el fichero que contiene la melodía final y su acompañante. Este método llama a su vez al paquete Player, el cual creará el MIDI de la melodía final.

6 Resultados experimentales

En este apartado se van a comentar los experimentos realizados y los resultados obtenidos en el proyecto. Estos experimentos se han realizado en 3 estilos distintos de música (Clásica, Metal y Blues), de modo que se pueda evaluar la capacidad del sistema de modelar patrones en estilos distintos.

Dentro de cada estilo musical se han realizado una serie de experimentos distintos, como es el crear una melodía a partir de un único patrón, crear una melodía a partir de varios patrones de duración media y el mismo caso con patrones de duración corta. Las combinaciones anteriormente mencionadas se han probado extrayendo patrones de una sola obra y extrayendo patrones de una selección de obras del estilo musical a modelar.

6.1 Música clásica

En este apartado se van a ver todos los experimentos realizados con partituras de música clásica. Al inicio de cada apartado dedicado a un estilo musical se va a incluir un resumen con los datos extraídos de los tries de frases de altura y duración a la hora de realizar los experimentos. Los datos de los tries de intervalos de altura y duración utilizados para los experimentos con una sola obra son los siguientes:

Número de obras	1
Número de frases de altura	150
Número de frases distintas de altura	93
Tamaño medio de frases	7 intervalos

Tabla 6 Datos extraídos del trie de intervalos de altura de La Marcha Turca

Número de obras	1
Número de frases de duración	150
Número de frases distintas de duración	62
Tamaño medio de frases	7 intervalos

Tabla 7 Datos extraídos del trie de intervalos de duración de La Marcha Turca

La obra utilizada ha sido La Marcha Turca y ha sido escogida debido a la presencia de patrones melódicos reconocibles y la poca variación de sus patrones rítmicos de duración. La partitura original de la obra es la siguiente:

Turkish March

W. A. Mozart

Words & Music by W. A. Mozart

$\text{♩} = 150$

Part A

Part A consists of five measures. Measure 1 is in 4/4 time, measure 2 is in 4/4 time, measure 3 is in 2/4 time, measure 4 is in 2/4 time, and measure 5 is in 2/4 time. The key signature has one sharp (F#).

Part B

Part B consists of 14 measures. Measures 6-13 are in 2/4 time, measure 14 is in 4/4 time, measure 15 is in 4/4 time, measure 16 is in 4/4 time, measure 17 is in 4/4 time, measure 18 is in 4/4 time, and measure 19 is in 4/4 time. The key signature has one sharp (F#).

Part C

Part C consists of 14 measures. Measures 20-24 are in 2/4 time, measure 25 is in 4/4 time, measure 26 is in 2/4 time, measure 27 is in 2/4 time, measure 28 is in 2/4 time, measure 29 is in 2/4 time, measure 30 is in 2/4 time, measure 31 is in 2/4 time, measure 32 is in 2/4 time, and measure 33 is in 4/4 time. The key signature has one sharp (F#).

Part D

Part D consists of 5 measures. Measures 34-38 are in 2/4 time. The key signature has one sharp (F#).

Part E

Part E consists of 16 measures. Measures 39-44 are in 2/4 time, measure 45 is in 4/4 time, measure 46 is in 4/4 time, measure 47 is in 4/4 time, measure 48 is in 4/4 time, measure 49 is in 4/4 time, measure 50 is in 4/4 time, measure 51 is in 4/4 time, measure 52 is in 4/4 time, measure 53 is in 2/4 time, measure 54 is in 2/4 time. The key signature has one sharp (F#).

Figura 42 Partitura original de La Marcha Turca

Experimento 0: Melodía formada por un solo patrón.

-Objetivos

Mediante este experimento se tratará de generar una melodía que contenga un único patrón con frases reconocibles de la partitura original.

-Preparación

Para la realización de este experimento se va a utilizar la Marcha Turca, obra compuesta por el compositor Mozart. Partiendo de la partitura de esta composición, se va a generar una melodía compuesta por un único patrón de larga duración, es decir, el resultado final se conseguirá mediante la unión de frases obtenidas de esta obra hasta alcanzar una duración equivalente a 140 negras.

-Resultados



Figura 43 Melodía generada a partir de La Marcha Turca

Como se puede apreciar en la partitura de la melodía, en el resultado obtenido no se aprecian grandes cambios respecto a las duraciones de las notas. Esto es debido al formato del experimento, realizado con un único patrón, ya que cada frase elegida para formar parte del patrón tiene como duración inicial la duración de la última nota de la frase anterior. Además, los patrones de intervalos de duración captados en el trie reflejan que en La Marcha Turca no hay apenas variaciones de duración, lo que posibilita estos pocos cambios en las figuras de las notas.

Al fijarnos en el compás 13 de la partitura de la obra generada se puede apreciar que la siguiente sección:



Figura 44 Fragmento de la melodía generada a partir de La Marcha Turca

Tiene gran semejanza con la siguiente sección de la partitura original:



Figura 45 Fragmento de La Marcha Turca

Se puede apreciar como la progresión de las duraciones es similar, a pesar de que las figuras no sean las mismas, ya que estas se generan de modo aleatorio. Respecto a la progresión de las notas, es lógico que difieran entre si, ya que para acompañar a esta frase de duraciones el programa escoge de forma aleatoria a cualquier frase que tenga la misma longitud, por ello este resultado.

También se aprecia en secciones como esta, donde se generan frases sin variaciones de duración y con una progresión de notas con intervalos pequeños.



Figura 46 Fragmento segundo de la melodía generada a partir de La Marcha Turca

Hay semejanzas con secciones de la obra original como la que se ve en la imagen. Se puede ver como se han incluido frases procedentes de esta sección para la generación de la melodía. La progresión de las notas es similar en ambos casos, aunque cabe decir que en la melodía que hemos obtenido esta progresión esta interpretada varias octavas más

alta que la original. Por último, aunque la progresión sea parecida, al escogerse de forma aleatoria la nota inicial, ambas secciones tienen semejanzas pero no por ellos suenan iguales.



Figura 47 Fragmento segundo de La Marcha Turca

Cabe decir que la posibilidad de incluir frases que contengan intervalos de duración más grandes durante la creación del patrón queda limitada por la duración inicial que se escoja (de forma aleatoria) a la hora de generar las duraciones, como es este caso. Si de forma aleatoria se escoge como duración inicial una figura de poca duración o de gran duración, se corre el riesgo de que las frases que contienen intervalos de duración muy grandes no sean válidas, por lo que en ese caso se tiende a escoger frases cuyas variaciones de duración sean más moderadas. Como podemos ver en la partitura, la Marcha Turca no es una composición con grandes intervalos de duración por lo que el resultado final puede resultar un tanto monótono en cuanto al aspecto rítmico.

Es reseñable que al tratarse de una melodía compuesta por un único patrón, carece de repetición de estructuras, por lo que la composición suena algo caótica al no dar sensación de naturalidad y coherencia. No obstante, si se aprecia como se han podido captar aspectos de la obra original en el resultado final, por lo que el objetivo de modelar una sola obra y lograr reconocer partes de esta en la composición obtenida puede darse por satisfecha.

Experimento 1: Melodía formada por varios patrones

-Objetivos

Para crear una melodía los compositores utilizan dos recursos básicos: la repetición y el contraste. La repetición de una idea musical, puede ser literal o presentar alguna variación con respecto a la idea original. La repetición ayuda a dar unidad a la obra musical (algunas frases pueden escucharse varias veces a lo largo de una obra). A su vez, el contraste evita que la música sea monótona, presentando de una idea nueva que contrasta con lo ya escuchado. En este experimento tratamos de introducir estos 2 elementos generando una melodía en la que haya repetición de patrones y diversidad de patrones para producir contraste.

El objetivo de este nuevo experimento es mejorar la musicalidad obtenida en el apartado anterior, evaluando el sentido de repetición y de contraste producido al incluir más patrones.

-Preparación

Para la realización de este experimento se van a generar varios patrones para construir una melodía a partir de ellos. En concreto se van a generar 2 patrones de 16 negras duración (patrones A y C) y 2 patrones más de duración 24 negras (patrones B y D). La estructura de estos patrones se muestra a continuación:

Patrón A



Figura 48 Patrón A Experimento 1

Patrón B



Figura 49 Patrón B Experimento 1

Patrón C



Figura 50 Patrón C Experimento 1

Patrón D



Figura 51 Patrón D Experimento 1

Se va a seguir la siguiente secuencia de patrones a la hora de crear la melodía:

ABABCABDCAB

De modo que la estructura principal la formen los patrones A-B y los patrones C y D sirvan como puente para volver a llegar a esta estructura.

- Resultados



Figura 52 Melodía obtenida Experimento 1

Se puede ver en la partitura la sucesión de patrones, de modo que esta unión forma la melodía. Visto el resultado obtenido, se puede comprobar que el generar una composición con estructuras que se repitan hace que mejore la sonoridad, debido a que hace que suene más natural y menos caótico que en el caso de generarla con un solo patrón.

También se aprecia una mayor dinámica al combinar patrones con figuras de mayor duración como el B con patrones con figuras de menor duración como el A. Esto hace que la composición tenga una variedad rítmica mas acentuada. Respecto a la sonoridad, el utilizar frases provenientes de una misma obra hace que el resultado sea más compacto, dando una mayor sensación de uniformidad. Sin embargo, al ser patrones de duración 24 y 16 negras respectivamente, propicia que en un mismo patrón se mezcle frases procedentes de distintas secciones de la obra original, lo que hace, especialmente en el patrón C, que sea algo menos agradable para el oído.

Cabe decir que la obtención de patrones que sonasen diferentes entre si ha sido una tarea costosa, ya que al tener que elegir entre frases de una misma obra el número de opciones es limitado.

Experimento 2: Melodía formada por varios patrones de duración corta.

-Objetivos

Una vez realizado el experimento del apartado anterior, se aprecia una mejora en la musicalidad al incluir patrones distintos. Sin embargo, la sensación de contraste es todavía elevada. Por ello, se va a reducir el tamaño de los patrones empleados para fomentar la sensación de repetición frente a la de contraste.

El objetivo de este experimento es evaluar si al reducir la longitud de los patrones se aumenta la sensación de repetición frente a la de contraste y por tanto las obras generadas tienen mayor musicalidad.

- Preparación

Para la realización de este experimento se van a generar varios patrones de duración corta para construir una melodía a partir de ellos, siendo la progresión de notas más parecida entre si al pertenecer a una misma obra.

En concreto se van a generar 4 patrones de 8 negras de duración. Se ha establecido esta duración ya que coincide con el tamaño máximo de las frases que se obtienen al parsear las partituras. No obstante, no se puede asegurar que un patrón siempre sea equivalente a una sola frase, ya que al parsear la frase original esta puede tener una duración de 8 negras pero al generarla con la aplicación las figuras que la compongan pueden ser distintas. Los patrones utilizados son los siguientes:

Patrón A



Figura 53 Patrón A Experimento 2

Patrón B



Figura 54 Patrón B Experimento 2

Patrón C



Figura 55 Patrón C Experimento 2

Patrón D



Figura 56 Patrón D Experimento 2

Se va a seguir la siguiente secuencia de patrones a la hora de crear la melodía:

ABABCD CAB CABCD CAB CAB

De modo que la estructura principal la formen los patrones A-B y los patrones C y D sirvan como puente para volver a llegar a esta estructura.

-Resultados



Figura 57 Melodía obtenida Experimento 2

Como se puede ver en la partitura, el combinar patrones de corta duración hace que se obtenga un resultado más agradable al oído al aumentar la sensación de contraste. La inclusión de patrones que se ajusten más a la duración original de una frase logra que cada patrón suene más natural, por lo que al unir los distintos patrones en una sola secuencia se logra un resultado más coherente y uniforme respecto a la sonoridad. A esto ayuda que no haya grandes cambios respecto a los intervalos de altura al pertenecer todas las frases a la misma obra y al acompañamiento que se ha generado para la obra, dando a esta secuencia un aspecto de melodía global.

Respecto a la secuencia rítmica, se eligieron patrones con sucesiones de figuras distintas entre sí para intentar dar un aspecto más heterogéneo a la composición. El resultado ha sido satisfactorio en el sentido de que la combinación de patrones más lentos con otros más rápidos da a la composición un tono más alegre y variado, sin que por ello se aprecie incoherencia entre ellos, aspecto a lo que ayuda el que solo haya frases de duración de una misma obra.

Cabe decir que al igual que en el experimento anterior, la obtención de patrones que cumplieren los requisitos para este experimento ha sido dificultosa, teniendo que realizar varias ejecuciones del programa hasta encontrar patrones con un sonido convincente y distinto entre sí. Esto es debido a que el experimento se ha realizado con una sola obra y un número bajo de frases distintas (unas 150). Al ser escogidos los patrones de forma aleatoria, la probabilidad de obtener patrones distintos es baja.

Experimento 3: Melodía formada con obras de distintos autores por patrones de duración corta

-Objetivos

A diferencia de los experimentos realizados anteriormente, en este se van a emplear partituras de varios autores, por lo que el número de frases disponibles para generar los patrones aumenta. Este hecho propicia que estén incluidas frases muy distintas entre sí a pesar de pertenecer al mismo estilo musical.

El objetivo de este experimento es evaluar la musicalidad obtenida al incluir frases de distintos autores dentro de una misma obra. Se va a observar especialmente como repercute el incluir frases de distintas obras en el sentido del contraste.

-Preparación

Los datos extraídos de los tries de intervalos de altura y duración utilizados para este experimento son los siguientes:

Número de obras	10
Número de frases de altura	2608
Número de frases distintas de altura	789
Tamaño medio de frases	23 intervalos

Tabla 8 Datos extraídos del trie de intervalos de altura con obras de Música Clásica

Número de obras	10
Número de frases de duración	2608
Número de frases distintas de duración	908
Tamaño medio de frases	23 intervalos

Tabla 9 Datos extraídos del trie de intervalos de duración con obras de Música Clásica

Para su realización se van a generar varios patrones de duración corta que servirán para construir una melodía a partir de ellos. En concreto se van a generar 4 patrones de 8 negras de duración. Se ha establecido esta duración ya que coincide con el tamaño máximo de las frases que se obtienen al parsear las partituras. Los patrones utilizados son los siguientes:

Patrón A



Figura 58 Patrón A Experimento 3

Patrón B



Figura 59 Patrón B Experimento 3

Patrón C



Figura 60 Patrón C Experimento 3

Patrón D



Figura 61 Patrón D Experimento 3

Al igual que en el experimento anterior, se han generado unos patrones diferentes entre sí, teniendo en cuenta que al tener mayor variedad de partituras este objetivo debe ser fácilmente alcanzable.

Se va a seguir la siguiente secuencia de patrones a la hora de crear la melodía

ABABABCD CABABCD CABABAAB

De modo que la estructura principal la formen los patrones A-B y los patrones C y D sirvan como puente para volver a llegar a esta estructura.

-Resultados



Figura 62 Melodía obtenida Experimento 3

El resultado obtenido se puede catalogar como agradable al oído y con un aspecto uniforme al añadirle su correspondiente acompañamiento. A pesar de estar compuesto por frases pertenecientes a distintas obras, la creación de patrones de 8 negras, ajustándose al tamaño máximo de las frases logra que se generen patrones breves y cuya unión no resulta caótica ni con un sentido del contraste demasiado elevado al pertenecer todas al mismo estilo musical.

Comparando el resultado obtenido con el experimento anterior se aprecia una mayor musicalidad y uniformidad en el experimento 2, lo que puede ser debido a que todos los patrones pertenecen a la misma obra. Sin embargo, tal como se comentó en el párrafo anterior, la melodía obtenida en el experimento 3 suena agradable al oído al pertenecer todos sus patrones a un mismo estilo. Es el hecho de incluir patrones de distintos autores lo que quizá potencia la sensación de contraste y hace que suene algo menos compacta. A esto ayuda el mayor número de distintas progresiones de intervalos de altura y duraciones incluidos al trabajar con varias obras.

Cabe decir que al igual que en el experimento anterior, la generación de patrones para realizar el experimento no ha sido dificultosa por el amplio número de frases disponibles.

Conclusiones Música Clásica

Una vez realizadas estos experimentos, se puede ver que donde mejor resultado se obtiene es en los experimentos con patrones de 8 negras, tanto para una sola obra como para todo un conjunto de obras de música clásica. Se aprecia una mayor conjunción y musicalidad en estos experimentos al generar patrones similares a los que se extrajeron de las partituras, especialmente en el caso de una sola obra.

Respecto a los experimentos con un solo patrón, resultan algo anárquicas por su falta de estructura y suenan por ello menos naturales por la falta de repetición, sin embargo en el experimento realizado para una sola obra si que se pueden apreciar patrones pertenecientes a la obra original.

En cuanto al resultado obtenido con patrones de duración media, su musicalidad es mayor respecto a experimentos con un solo patrón. Sin embargo, la sensación de contraste es elevada respecto a los experimentos con patrones de 8 negras, siendo estos los experimentos con resultados más satisfactorios y de mayor musicalidad

Profundizando en este último punto, señalar que se han realizado experimentos con varios patrones de duración media y formados por un solo patrón con obras de distintos autores de música clásica. Sin embargo, no han sido incluidos en esta memoria debido que el resultado obtenido es poco destacable, ya que la musicalidad obtenida con patrones de 8 negras es superior. Estos experimentos están incluidos como apéndices al final de la memoria.

Cabe decir que aunque se hayan generado patrones y frases procedentes de distintas obras, su unión en ningún caso ha resultado extraña al oído, ya que se ha modelado un estilo musical con estructuras y progresiones en común.

Para finalizar, se va a proceder a realizar experimentos con patrones de 8 negras en otros estilos musicales, de modo que se compruebe que se puede trasladar el proceso efectuado a otros estilos.

6.2 Metal

El objetivo del conjunto de experimentos que se van a hacer con el estilo musical Metal es comprobar que el proceso realizado con obras pertenecientes a música clásica es aplicable a otros estilos. Por ello se realizarán experimentos con patrones cortos para una sola obra y para un conjunto de obras de este estilo, de modo que se pueda observar que los resultados que se obtienen se adecuan al estilo modelado.

Los datos extraídos de los tries de intervalos de altura y duración para hacer los experimentos con una sola obra son las siguientes:

Número de obras	1
Número de frases de altura	197
Número de frases distintas de altura	93
Tamaño medio de frases	15 intervalos

Tabla 10 Datos extraídos del trie de intervalos de altura de la obra Davidian

Número de obras	1
Número de frases de duración	197
Número de frases distintas de duración	57
Tamaño medio de frases	15 intervalos

Tabla 11 Datos extraídos del trie de intervalos de duración de la obra Davidian

Y los datos extraídos de los tries de intervalos de altura y duración a la hora de realizar el experimento con todas las obras son estos:

Número de obras	10
Número de frases de altura	3457
Número de frases distintas de altura	617
Tamaño medio de frases	7 intervalos

Tabla 12 Datos extraídos del trie de intervalos de altura con obras de Metal

Número de obras	10
Número de frases de duración	3457
Número de frases distintas de duración	765
Tamaño medio de frases	7 intervalos

Tabla 13 Datos extraídos del trie de intervalos de duración con obras de Metal

Experimento 4: Melodía formada por patrones de duración corta obtenidos de una sola obra.

-Objetivos

El objetivo de este experimento es crear una melodía basándose en patrones generados a partir de una misma obra, de modo que en el resultado final se aprecien semejanzas con la obra original.

-Preparación

Para realizar este experimento se van a generar 4 patrones de 8 negras de duración, al igual que en los experimentos de igual criterio que se realizaron anteriormente. Los patrones utilizados son los siguientes:

Patrón A



Figura 63 Patrón A Experimento 4

Patrón B



Figura 64 Patrón B Experimento 4

Patrón C



Figura 65 Patrón C Experimento 4

Patrón D



Figura 66 Patrón D Experimento 4

Se va a seguir la siguiente secuencia de patrones a la hora de crear la melodía

ABABABCD CABABCD CABABAAB

De modo que la estructura principal la formen los patrones A-B y los patrones C y D sirvan como puente para volver a llegar a esta estructura.

-Resultados



Figura 67 Melodía obtenida Experimento 4

El resultado obtenido se puede ver en la partitura. Lo primero que se aprecia es un cambio en la sonoridad bastante notable respecto a los experimentos realizados con música clásica. La melodía se compone de figuras de menor duración y abundan los intervalos de altura unísonos acordes al estilo Metal, como se puede ver en los primeros compases de la obra.

A lo largo de la obra generada se pueden apreciar secciones similares a las que se pueden hallar en la obra original como las siguientes:

- En este caso, vemos como tanto las duraciones como las notas llevan una progresión similar en ambas partituras.



Figura 68 Fragmento de la partitura de Davidian



Figura 69 Fragmento de la Melodía obtenida en Experimento 4

- En este caso tenemos, vemos como se refleja esta progresión de duraciones en la obra creada, siendo distinta la progresión de la altura al haberse escogido una frase diferente.



Figura 70 Fragmento de la partitura de Davidian



Figura 71 Fragmento de la Melodía obtenida en Experimento 4

Respecto a su sonoridad, la unión de estos patrones suena coherente y compacta a lo largo de la melodía, motivado sobre todo por pertenecer a una misma obra.

Por último, al igual que con el experimento del mismo tipo realizado con música clásica, destacar que la generación de los patrones para el experimento ha sido algo dificultoso, teniendo que realizar muchas ejecuciones hasta encontrar un patrón adecuado. Esto es debido a la muestra escasa de frases obtenidas de una sola obra (unas 200), con lo que la probabilidad de generar patrones distintos es baja.

Experimento 5: Melodía formada por patrones de duración corta obtenidos de varias obras.

-Objetivos

Este experimento es similar al realizado anteriormente, con la distinción de que se va a realizar con obras de distintos autores. Debido a ello, el objetivo del experimento es comprobar el resultado sonoro de crear una melodía con frases de distintos autores del estilo metal.

-Preparación

Los patrones generados para el experimento son los siguientes:

Patrón A



Figura 72 Patrón A Experimento 5

Patrón B



Figura 73 Patrón B Experimento 5

Patrón C



Figura 74 Patrón C Experimento 5

Patrón D



Figura 75 Patrón D Experimento 5

Y la secuencia de patrones será la siguiente:

ABABABCD CABABCD CABABAAB

De modo que la estructura principal la formen los patrones A-B y los patrones C y D sirvan como puente para volver a llegar a esta estructura.

-Resultados

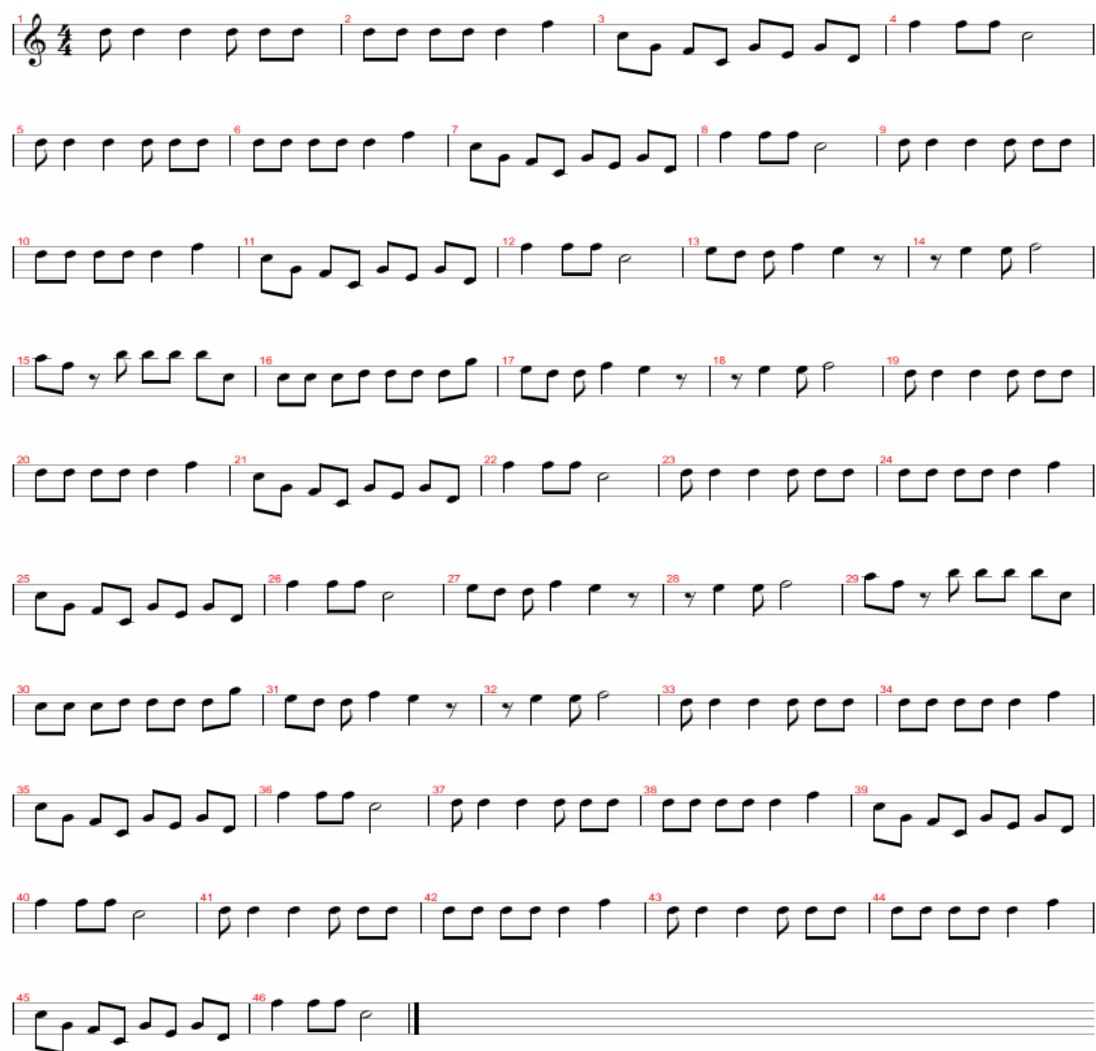


Figura 76 Melodía obtenida Experimento 5

El resultado obtenido suena natural y coherente, pese a generarse a partir de distintas obras, a lo que ayuda la repetición de las estructuras dándole a la melodía un sentido de la repetición más elevado.

Es destacable que la realización de este experimento ha sido más fácil ya que se disponían de unas 3400 frases para elegir a la hora de generar los patrones.

Conclusiones Metal

Comparando los experimentos 4 y 5, se aprecia que el experimento con una sola obra suena mejor y es más uniforme, al igual que pasaba con el experimento de Música Clásica. Esto es debido a que todos los patrones pertenecen a una misma obra, aunque si se echa en falta algo más de variación en los intervalos de altura, ya que la obra empleada tiende a emplear muchos intervalos unísonos. Sin embargo, en el experimento 5 el resultado es menos homogéneo, ya que hay progresiones de notas más variadas y más cambios en los intervalos de duración producto del uso de varias obras para crear los patrones. A pesar de ello, esta melodía suena agradable al oído y la concatenación de los patrones no desentona.

En general, los resultados obtenidos son satisfactorios en el sentido de que se han podido generar melodías que suenan similares al estilo modelado, por lo que se puede asegurar que el proceso llevado a cabo por el programa se ha podido aplicar correctamente.

6.3 Blues

El objetivo del conjunto de experimentos que se van a hacer con el estilo musical Blues es comprobar que el proceso realizado con obras pertenecientes a música clásica es aplicable a otros estilos. Por ello se realizarán experimentos con patrones cortos para una sola obra y para un conjunto de obras de este estilo, de modo que se pueda observar que los resultados que se obtienen se adecuan al estilo modelado.

Los datos extraídos de los tries de intervalos de altura y duración utilizados para hacer los experimentos con una sola obra son los siguientes:

Número de obras	1
Número de frases de altura	197
Número de frases distintas de altura	93
Tamaño medio de frases	15 intervalos

Tabla 14 Datos extraídos del trie de intervalos de altura de la obra Take Five

Número de obras	1
Número de frases de duración	197
Número de frases distintas de duración	57
Tamaño medio de frases	15 intervalos

Tabla 15 Datos extraídos del trie de intervalos de duración de la obra Take Five

Los datos extraídos de los tries de intervalos de altura y duración utilizados para el experimento con varias obras son:

Número de obras	1
Número de frases de altura	237
Número de frases distintas de altura	106
Tamaño medio de frases	3 intervalos

Tabla 16 Datos extraídos del trie de intervalos de altura de obras de Blues

Número de obras	1
Número de frases de duración	237
Número de frases distintas de duración	123
Tamaño medio de frases	3 intervalos

Tabla 17 Datos extraídos del trie de intervalos de duración con obras de Blues

Experimento 6: Melodía formada por patrones de duración corta obtenidos de una sola obra.

-Objetivos

El objetivo de este experimento es crear una melodía a partir de patrones generados a partir de una misma obra, de modo que en el resultado final se aprecien semejanzas con la obra original.

-Preparación

Los patrones generados son los siguientes:

Patrón A



Figura 77 Patrón a Experimento 6

Patrón B



Figura 78 Patrón B Experimento 6

Patrón C



Figura 79 Patrón C Experimento 6

Patrón D



Figura 80 Patrón D Experimento 6

Se va a seguir la siguiente secuencia de patrones a la hora de crear la melodía

ABABABCD CABABCD CABABAAB

De modo que la estructura principal la formen los patrones A-B y los patrones C y D sirvan como puente para volver a llegar a esta estructura.

-Resultados



Figura 81 Melodía obtenida en Experimento 6

Al tratarse de un estilo completamente distinto a los utilizados anteriormente, el resultado llama la atención por utilizar figuras de poca duración y una gran variación de notas. Al escucharse la melodía se puede apreciar que los fraseos que se han originado suenan similares entre si al pertenecer todos a una misma obra.

Dentro del midi generado, se pueden apreciar secciones semejantes a la obra original, como los siguientes:

- En este caso se observa como la progresión de las notas y las duraciones es similar, con la diferencia que se ha generado con figuras de mayor duración.



Figura 82 Fragmento de la partitura de Take Five



Figura 83 Fragmento de la Melodía generada en Experimento 6

- Al igual que en el ejemplo anterior, podemos ver como coinciden estas dos secciones de las 2 obras, con la diferencia de tocarse en una octava más baja en el midi generado.



Figura 84 Fragmento de la Melodía generada en Experimento 6



Figura 85 Fragmento de la partitura de Take Five

Por último, decir que la generación de los patrones para este experimento ha sido menos dificultosa respecto a otros estilos. Esto es debido a que a pesar de ser una sola obra, en este género hay poca repetición de estructuras, por lo que aumenta el número de frases distintas.

Experimento 7: Melodía formada por patrones de duración corta obtenidos de varias obras.

-Objetivos

Este experimento es similar a la realizada anteriormente, con la distinción de que se va a realizar con obras de distintos autores. Debido a ello, el objetivo del experimento es comprobar el resultado sonoro de crear una melodía con frases de distintos autores del estilo Blues.

-Preparación

Los patrones generados son los siguientes:

Patrón A



Figura 86 Patrón A Experimento 7

Patrón B



Figura 87 Patrón C Experimento 7

Patrón C



Figura 88 Patrón C Experimento 7

Patrón D



Figura 89 Patrón D Experimento 7

Y la estructura que se va a seguir a la hora de crear la melodía es la siguiente:

ABABABCD CABCDCABAAB

-Resultados:



Figura 90 Melodía obtenida en Experimento 7

Al escuchar el resultado, se observa que las distintas partes suenan más distintas entre si que en el experimento realizado para una sola obra. También se puede apreciar que la composición tiene figuras de poca duración y mucha variación de notas, habiendo pocos intervalos unísonos. Estas características se ajustan al sonido característico del Blues.

Por último, al igual que en todos los experimentos realizados con muchas obras, la generación de los patrones ha sido una tarea rápida.

Conclusiones Blues

Al igual que con el Metal, los resultados genéricos han sido satisfactorios, obteniendo unos midis con características del Blues. De este modo se comprueba que se puede aplicar el modelado de patrones a este estilo. Es reseñable que la comparación entre los experimentos 6 y 7 sigue los mismos parámetros que con los estilos anteriormente modelados, donde el experimento con patrones de una sola obra tiene un resultado más musical y suena más uniforme que con la mezcla de patrones de varias obras.

7 Conclusiones

Al establecer el objetivo principal del proyecto, se especificaron una serie de subobjetivos los cuales son necesarios cumplir en un grado suficiente de modo que se alcance el objetivo principal. En general el resultado obtenido por cada subobjetivo ha sido satisfactorio, no obstante a continuación se incluye una valoración de cada uno de ellos:

- Construcción de la Base de datos MusicXML:

El grado de cumplimiento de este subobjetivo es prácticamente total, ya que se ha conseguido recopilar una selección representativa de MusicXML de cada estilo musical a modelar. Esta tarea ha sido relativamente sencilla debido a que hay disponibles multitud de partituras MusicXML de cualquier estilo entre las que elegir para trabajar con ellas. Sin embargo, es destacable que dentro de estas partituras se ha trabajado con aquellas que tenían el formato adecuado, es decir, partituras que incluían una línea melódica sin utilizar acordes. Sería interesante que a la hora de recopilar los MusicXML se pudiese utilizar cualquiera, esta cuestión es tratada en las conclusiones del subobjetivo 'Extracción del conjunto de Patrones melódicos'.

- Desarrollo de una librería para el manejo de tries:

En general el cumplimiento de este subobjetivo es alto, al haber podido crear una librería que es capaz de insertar claves en la estructura de datos trie, obteniendo como salida un trie con las frases ordenadas por orden alfabético y con el número de veces que aparece cada frase asociado. Sin embargo, no se ha incluido el número de apariciones de los nodos internos, es decir, si tenemos la frase “Aba” por ejemplo, saber el número de veces que se recorre en el trie el camino formado por los intervalos “Ab”. Incluir esta opción implica añadir más información sobre los patrones que sigue un determinado estilo, por lo que podría ampliar la información obtenida de cada partitura. Al principio del desarrollo de la librería se valoró su inclusión, pero dificultades en su implementación hicieron que se descartara.

- Extracción del conjunto de Patrones melódicos:

Respecto a este subobjetivo, si bien es cierto que se ha conseguido extraer la información de los MusicXML, ha sido con una serie de restricciones a la hora de hacer la recopilación de MusicXML. Estas partituras deben referirse a una única línea melódica, no deben incluir figuras como la fusa, la semifusa o los tresillos, además de evitar el uso de acordes ya que el algoritmo creado para la extracción no contempla estos casos. Más allá de esta apreciación, si como entrada la aplicación tiene una partitura bien formada, es capaz de extraer patrones melódicos.

- Composición de Melodía con los patrones melódicos extraídos:

Dentro de este subobjetivo, cabe decir que al igual que en el subobjetivo anterior, este ha sido cumplido dentro de una serie de restricciones. Entre ellas no se ha tratado a la hora de generar nuevas melodías la inclusión de fusas y semifusas, la garantía de que las melodías cumplan las leyes armónicas y la utilización de bemoles y sostenidos. Por otro lado, tampoco se permite generar melodías con distintas armaduras (5x4, 12x8, 4x4, etc) ni con distintos acompañamientos. No obstante, tal y como se ha podido ver en los experimentos, la aplicación es capaz de generar nuevas melodías a partir de otros patrones melódicos.

- Implementación de Interfaz gráfica.

A pesar de no ser el objetivo principal de la aplicación el crear una GUI, si que es fundamental ofrecer al usuario una interfaz gráfica que muestre todas las posibilidades que ofrece la aplicación y facilite su uso. Partiendo de aquí, se ha creado una interfaz sencilla que da al usuario la opción de escoger el estilo musical a modelar, así como el crear patrones, escucharlos y elegir como combinarlos para crear una melodía. Por otro lado, no se han incluido opciones tales como escuchar directamente el MIDI generado o disponer de una opción para elegir que instrumento interpretará el acompañamiento, que podrían dotar al interfaz de un aspecto más sofisticado. No obstante, el cometido principal de un interfaz lo cumple, incluyendo todas las opciones fundamentales.

Con todos los subobjetivos satisfechos en mayor o menor medida se puede valorar el cumplimiento del objetivo principal de la aplicación, el cual consiste en extraer patrones reconocibles de un estilo musical con el fin de crear melodías pertenecientes a dicho estilo a partir de ellas.

Tras realizar experimentos con 3 estilos distintos (música clásica, Metal y Blues) se puede observar que las melodías obtenidas son distintas entre sí, apreciándose en cada una de ellas características del estilo en que se basan. Así se aprecia en cada estilo:

- Música Clásica: Aunque las características de este estilo dependen de muchos factores, si se aprecia en los resultados una sonoridad que recuerda a composiciones clásicas.
- Metal: Se ven muchos intervalos unísonos y figuras de poca duración que plasman los ritmos típicos de este estilo.
- Blues: Se aprecian estructuras con fraseos muy variados y grandes cambios en los intervalos de altura y duración. Estas características son típicas del blues, especialmente de las guitarras solistas que suelen incluir sus composiciones.

7.1 Líneas futuras

A la hora de añadir posibles mejoras y nuevas líneas de trabajo, dentro del desarrollo del módulo Trie se puede valorar modificar el código de inserción de claves dentro de la estructura para incluir no solo el número de apariciones de la clave completa (nodos hoja), sino también de los elementos que la forman (nodos internos). De este modo aumentaría la información respecto a las progresiones de intervalos utilizados en cada estilo, creándose melodías más representativas de cada estilo a partir de ellas.

Desde el punto de vista de posibles mejoras dentro del módulo de extracción de información de las partituras, sería interesante que las partituras MusicXML utilizadas puedan incluir acordes, fusas y semifusas, por lo que habría que modificar el algoritmo de extracción de información para incluir estas opciones. De este modo el abanico de partituras a poder utilizar se ampliaría y su contenido sería más heterogéneo.

En cuanto a la generación de melodías a partir de la información de los patrones extraídos, una línea futura puede ser el incluir una sección rítmica como parte del acompañamiento, además de modificar el comportamiento de los acordes que sirven de acompañamiento para que no suenen tan repetitivos al inicio de cada compás. Otras posibles opciones podrían ser el incluir acompañamientos más ricos, compuestos de varias líneas melódicas e interpretadas por distintos instrumentos, donde la elección de estos dependería del estilo escogido. Modificando los métodos encargados de crear el acompañamiento se podría incluir estas opciones para dotar a las melodías generadas de mayor dinamismo.

Siguiendo esta línea de enriquecer las melodías obtenidas, otra línea de trabajo sería incluir bemoles y sostenidos. Dentro del algoritmo de generación de melodías, habría que incluir un método que controlase cuando es adecuado incluir bemoles y/o sostenidos, dependiendo de la tonalidad en que se esté realizando la composición. Efectuando estos cambios aumentaría la musicalidad de las melodías obtenidas.

Por último, la interfaz de usuario cumple su cometido, pero se podrían desarrollar más funcionalidades que enriquecieran el resultado final. Posibles nuevas líneas pueden ser incluir una opción para elegir el instrumento que va a interpretar el acompañamiento, de modo que el usuario decida en función de sus gustos, o incluir una opción para reproducir directamente el MIDI generado. Para incluir estas opciones bastaría con modificar el código del interfaz, creando nuevos botones y asociándoles la nueva funcionalidad.

Respecto a la posible reutilización de los módulos implementados en futuros PFCs, el módulo Trie si podría ser utilizado en su mayoría en aquellos proyectos en los cuales sea necesario una estructura de este estilo para trabajar y obtener información de claves, de forma análoga a la realizada en este proyecto. El módulo Extraer sería aprovechable en posibles ampliaciones del presente PFC, como las líneas futuras mencionadas en este apartado, sirviendo de punto de partida para esos trabajos. Además, sería interesante que una parte de este módulo se reutilizara en futuros PFCs de informática y música. Nos referimos a la parte que convierte la partitura de una melodía en MusicXML a secuencias de intervalos de altura y de duración. Dentro de la posibilidad de ser utilizado para posibles ampliaciones de este PFC, nos encontramos con el módulo Interfaz,

estrechamente relacionado con el módulo Extraer y su funcionamiento. Por último, el módulo Player ha sido reutilizado de otro PFC, por lo que claramente es válido para multitud de trabajos que abarquen la generación y reproducción de MIDI.

Referencias

[BOE] *Boletín Oficial del Estado, XVI convenio colectivo estatal de empresas de consultoría y estudios de mercado*. Disponible [Internet]:
<<http://www.boe.es/boe/dias/2009/04/04/pdfs/BOE-A-2009-5688.pdf>> [26 de Septiembre de 2011].

[Bha] R. López de Mántaras, J. L. Arcos, *AI and Music. From Composition to Expressive Performance*, 2006, AI Magazine, 3-3.

[Cop] R. López de Mántaras, J. L. Arcos, *AI and Music. From Composition to Expressive Performance*, 2006, AI Magazine, 4-4.

[Cra] *¿Qué es el Modelo Vista-Controlador?*, Disponible [Internet]:
<<http://craftyman.net/mvc-en-php/>> [26 de Septiembre de 2011].

[Css] *Guía mensajes midi*, Disponible [Internet]: < www.css-audiovisual.com/areas/guias/midi-mensajes.htm > [26 de Septiembre de 2011].

[Dav] Dave Smith, *Digital standard for musical instruments*, Audio Engineering Society Congress, New York, 1981.

[Ebc] R. López de Mántaras, J. L. Arcos, *AI and Music. From Composition to Expressive Performance*, 2006, AI Magazine, 3-3.

[Exp] K. Verbeurgt, M. Dinolfo, M. Fayer, *Extracting Patterns in Music for Composition via Markov Chains*, 2004, Department of Computer Science State University N.York, 3-6.

[Feu] R. López de Mántaras, J. L. Arcos, *AI and Music. From Composition to Expressive Performance*, 2006, AI Magazine, 3-3.

[Hil] K. Verbeurgt, M. Dinolfo, M. Fayer, *Extracting Patterns in Music for Composition via Markov Chains*, 2004, Department of Computer Science State University N.York, 2-3.

[Jav] *Java Sound API Programmer's Guide*, Disponible [Internet]:
<<http://java.sun.com/j2se/1.3/pdf/javasound.pdf>> [26 de Septiembre de 2011].

[Lev] R. López de Mántaras, J. L. Arcos, *AI and Music. From Composition to Expressive Performance*, 2006, AI Magazine, 2-2.

[Ltm] *La Textura Musical*, Disponible [Internet]:
<[tps://sites.google.com/a/martinhalaja.com/la-textura-musical/recursos](https://sites.google.com/a/martinhalaja.com/la-textura-musical/recursos)> [26 de Septiembre de 2011].

[Min] R. López de Mántaras, J. L. Arcos, *AI and Music. From Composition to Expressive Performance*, 2006, AI Magazine, 3-3.

[Mms] R. López de Mántaras, J. L. Arcos, *AI and Music. From Composition to Expressive Performance*, 2006, AI Magazine, 4-4.

[Moo] R. López de Mántaras, J. L. Arcos, *AI and Music. From Composition to Expressive Performance*, 2006, AI Magazine, 2-2.

[Mus] *Teoría musical*, Disponible [Internet]: <<http://www.teoria.com>> [26 de Septiembre de 2011].

[Par] R. López de Mántaras, J. L. Arcos, *AI and Music. From Composition to Expressive Performance*, 2006, AI Magazine, 3-3.

[Rad] R. López de Mántaras, J. L. Arcos, *AI and Music. From Composition to Expressive Performance*, 2006, AI Magazine, 2-2.

[Rot] R. López de Mántaras, J. L. Arcos, *AI and Music. From Composition to Expressive Performance*, 2006, AI Magazine, 3-3.

[Sal] R. López de Mántaras, J. L. Arcos, *AI and Music. From Composition to Expressive Performance*, 2006, AI Magazine, 3-3.

[Sax] *Armonía musical*, Disponible [Internet]:
<<http://www.saxoargentina.com.ar/2008/08/08/armonia-capitulo-i>> [26 de Septiembre de 2011].

[Sch] R. López de Mántaras, J. L. Arcos, *AI and Music. From Composition to Expressive Performance*, 2006, AI Magazine, 4-4.

[Sim] R. López de Mántaras, J. L. Arcos, *AI and Music. From Composition to Expressive Performance*, 2006, AI Magazine, 3-3.

[Swi] *Creating a GUI With JFC/Swing*, Disponible [Internet]:
<<http://download.oracle.com/javase/tutorial/uiswing>> [26 de Septiembre de 2011].

[Wik] *Wikipedia, Enciclopedia de contenido libre*, Disponible [Internet]:
<www.wikipedia.org> [26 de Septiembre de 2011].

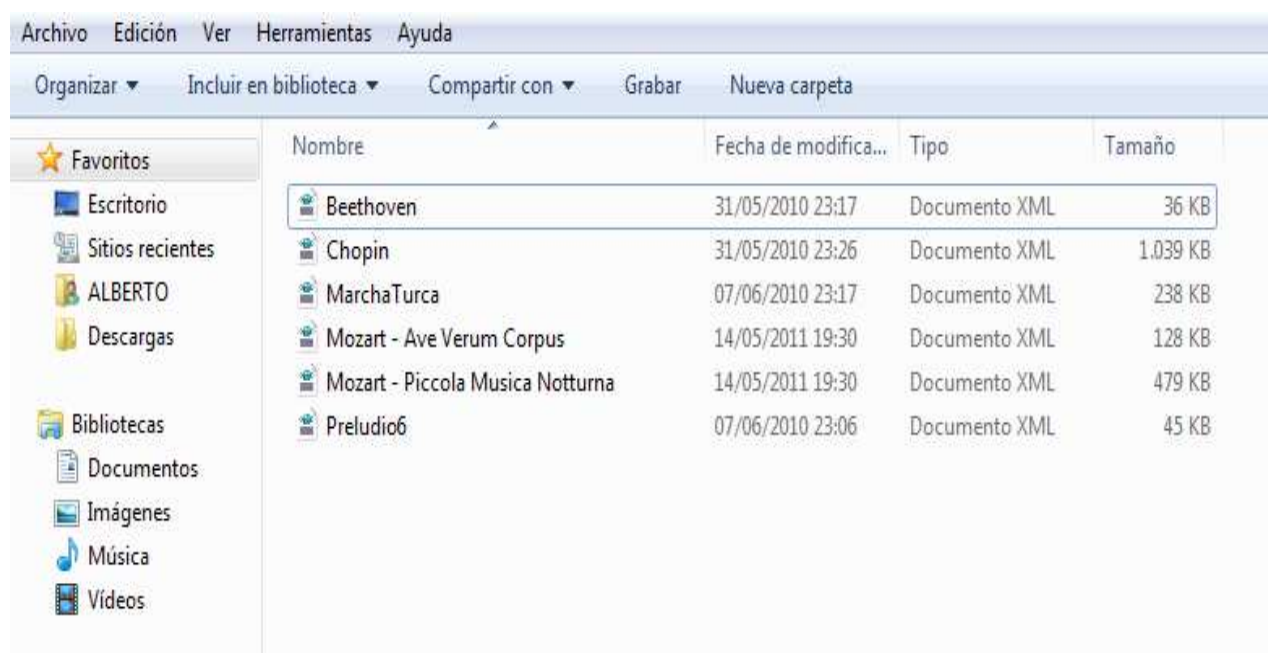
Apéndice 1: Manual de Usuario

En este apartado vamos a detallar los pasos a seguir para ejecutar la aplicación desarrollada en un sistema operativo Windows y así generar una melodía a partir de los patrones modelados. Cabe decir que deben seguirse estos mismos pasos en caso de querer ejecutar la aplicación en un sistema operativo Linux.

Requisitos para la ejecución

Para una correcta ejecución es necesario trabajar con un sistema operativo Windows Xp o superior, o en su defecto una distribución GNU/Linux. Además es necesario tener instalado el JDK 1.6 o superior de java.

Esta prueba se va a realizar en un ordenador personal con Windows 7 y JDK 1.6 para generar una melodía de música clásica a partir de unas partituras de MusicXML previamente seleccionadas. Para su ejecución basta con bajarse la aplicación, descomprimirla en el escritorio y ejecutar los pasos que se ven a continuación a través de la consola de comandos.



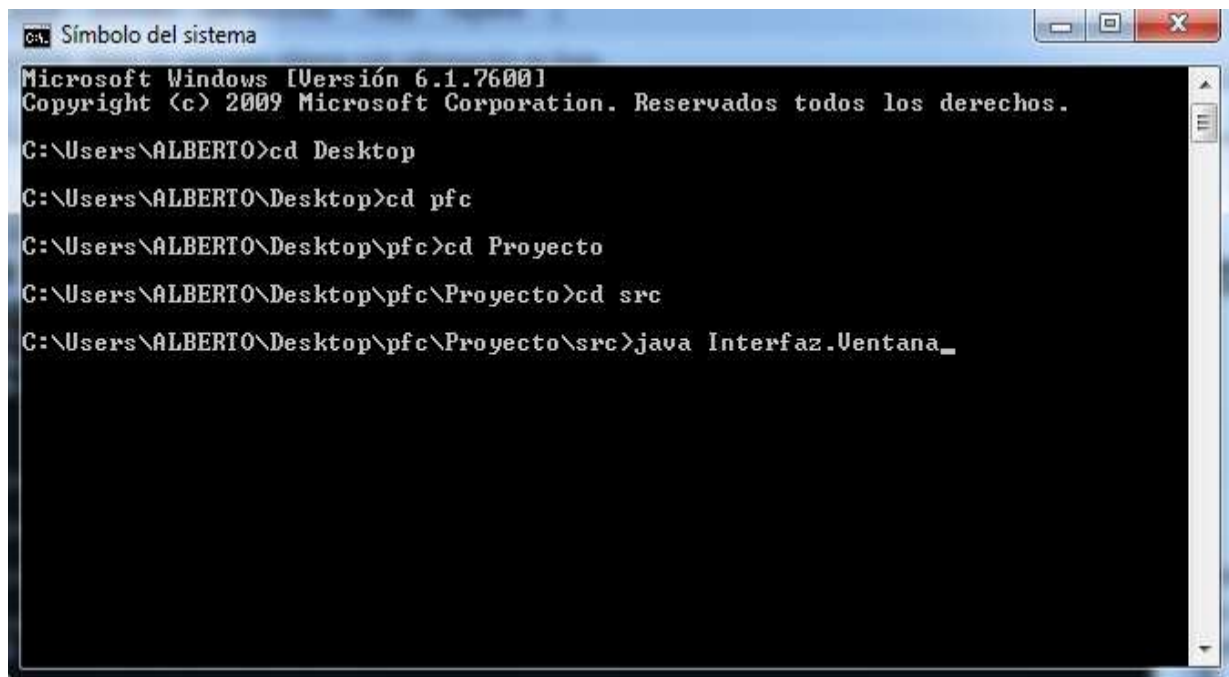
Nombre	Fecha de modifica...	Tipo	Tamaño
Beethoven	31/05/2010 23:17	Documento XML	36 KB
Chopin	31/05/2010 23:26	Documento XML	1.039 KB
MarchaTurca	07/06/2010 23:17	Documento XML	238 KB
Mozart - Ave Verum Corpus	14/05/2011 19:30	Documento XML	128 KB
Mozart - Piccola Musica Notturna	14/05/2011 19:30	Documento XML	479 KB
Preludio6	07/06/2010 23:06	Documento XML	45 KB

Figura 91 Partituras MusicXml utilizadas

Pasos necesarios para ejecutar

Lo primero que hay que hacer es abrir una consola de comandos, para ello vamos al botón de inicio→Todos los programas→Accesorios→Símbolo del sistema. De este modo accedemos a la consola de comandos, nos ubicamos en el escritorio y localizamos la carpeta que contiene la aplicación, llamada pfc.

Una vez localizada, el siguiente paso es ejecutar la aplicación, para ello tenemos que seguir la ruta siguiente desde el escritorio: cd pfc → cd Proyecto → cd src. Ubicados en la carpeta src, es necesario escribir el comando “java Interfaz.Ventana_”, de modo que se invoque el interfaz de usuario de la aplicación.



```
Microsoft Windows [Versión 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\ALBERTO>cd Desktop
C:\Users\ALBERTO\Desktop>cd pfc
C:\Users\ALBERTO\Desktop\pfc>cd Proyecto
C:\Users\ALBERTO\Desktop\pfc\Proyecto>cd src
C:\Users\ALBERTO\Desktop\pfc\Proyecto\src>java Interfaz.Ventana_
```

Figura 92 Pasos para ejecutar

De este modo se completan todos los pasos necesarios para utilizar la aplicación.

Manejo de la aplicación

Al invocar el interfaz de usuario, se presenta una pantalla con una serie de opciones tal y como se ve en la imagen siguiente:

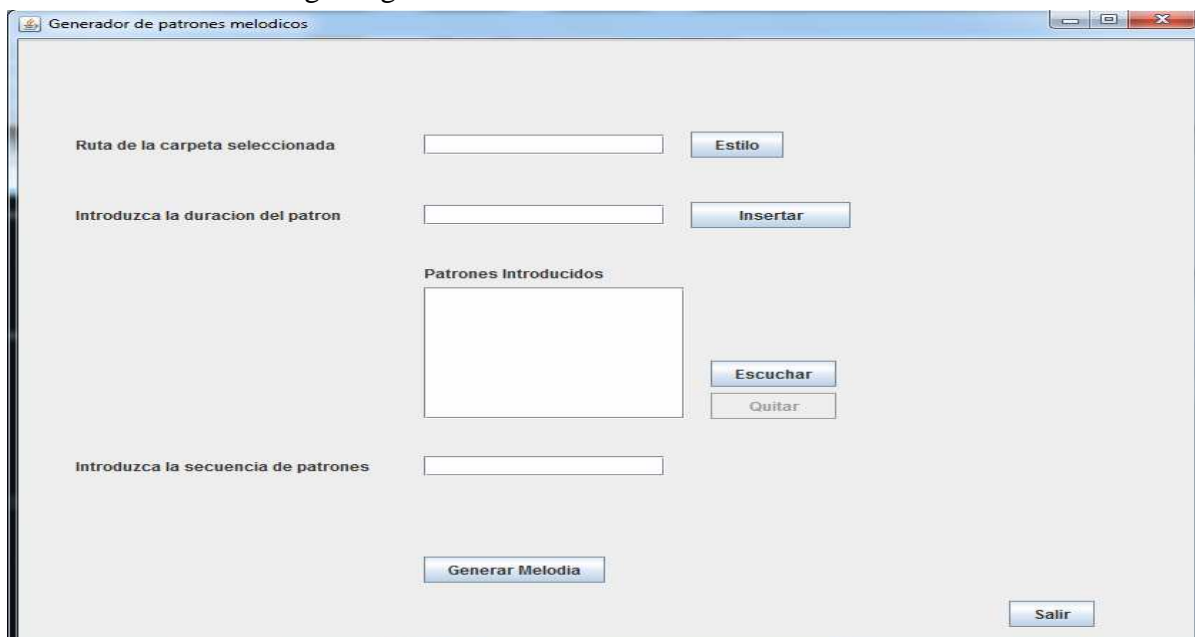


Figura 93 Interfaz de usuario

El primer paso es elegir el estilo musical a modelar mediante el botón estilo, el cual despliega un menú que nos permite elegir la carpeta con las partituras del estilo con el que vamos a trabajar. Cabe decir que es necesario incluir dentro de la carpeta seleccionada solo partituras MusicXML del estilo a analizar. En este caso el estilo seleccionado es clásica, como se ve en la imagen:

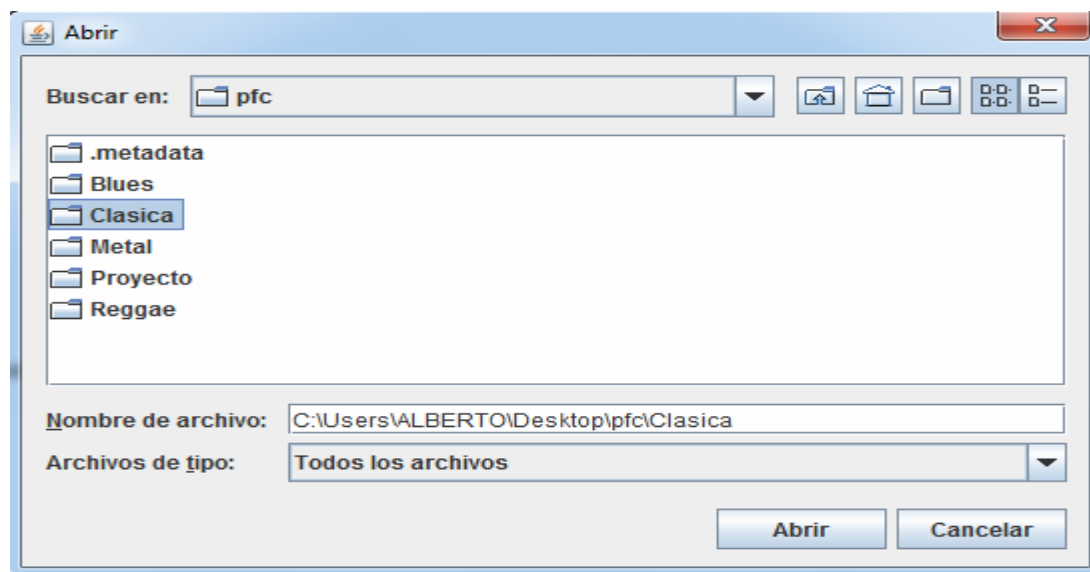


Figura 94 Elección de estilo

A continuación se crean los patrones musicales. Para ello se introduce la duración en negras que se quiere que tenga un patrón en el campo a la izquierda del botón insertar (50, 80, 150, etc.) y se pulsa el botón insertar. Hecho esto, los patrones creados se insertan en la lista de patrones que figura en el centro de la interfaz, como se puede ver a continuación.

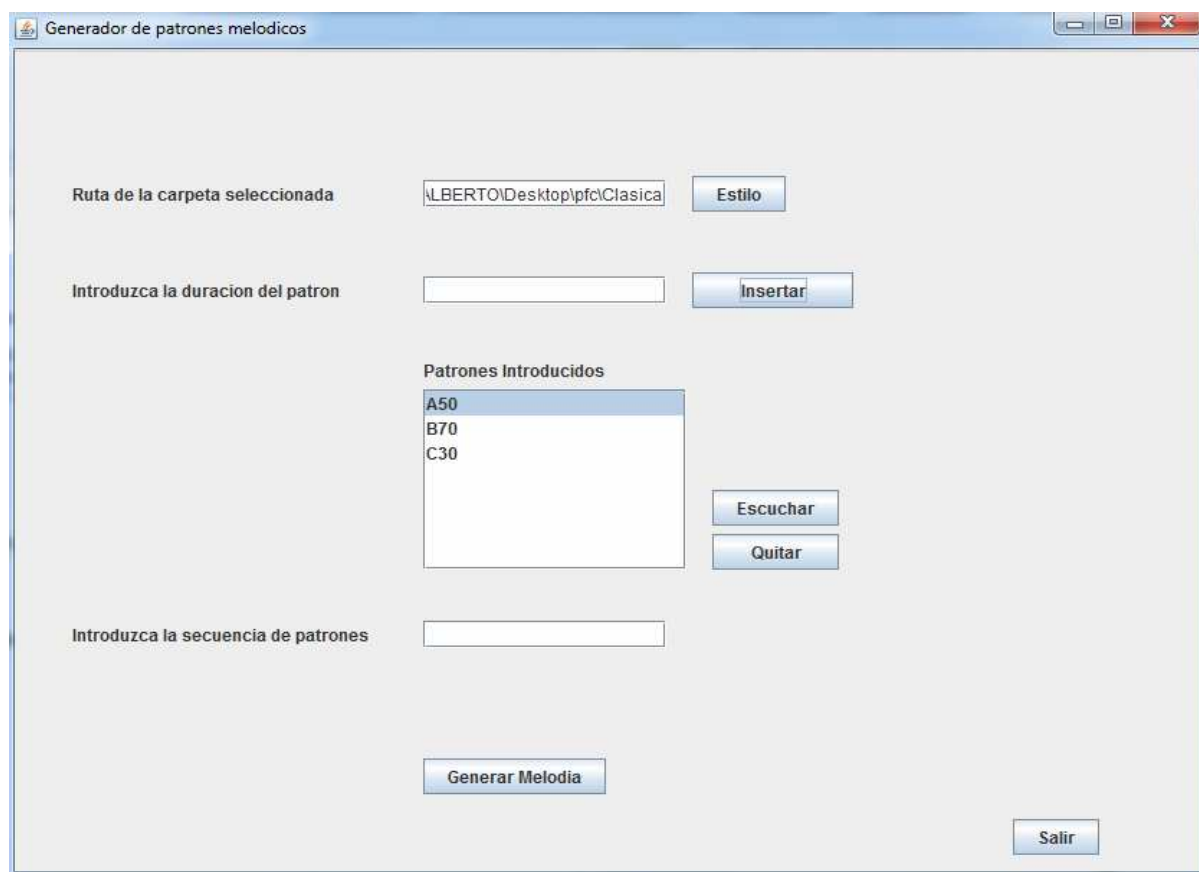


Figura 95 Interfaz con patrones ya creados

Como se puede ver, en esta demostración se han creado 3 patrones, A50, B70 y C30. Haciendo clic encima de ellos y pulsando el botón Escuchar, se puede oír el MIDI correspondiente al patrón creado, de modo que se pueda decidir si este patrón se incluye o no en la melodía final en función de si le gusta como suena al usuario. En caso de que el resultado del patrón no le guste, el usuario puede eliminarlo de la lista pulsando el botón Quitar mientras tiene seleccionado el patrón a eliminar.

Una vez el usuario tenga decididos los patrones que quiere formen la melodía, lo siguiente que tiene que hacer es elegir el orden en el que van a aparecer. En este caso los 3 patrones generados son del gusto del usuario, por lo que la secuencia que se va a generar está formada por A50 + B70 + C30 + A50 + C30 + B70. Para llevarlo a cabo basta con hacer doble clic sobre el elemento de la lista que se quiere agregar a la secuencia de patrones. En caso de querer eliminar algún elemento de la lista, se puede hacer manualmente sin ningún tipo de problema.

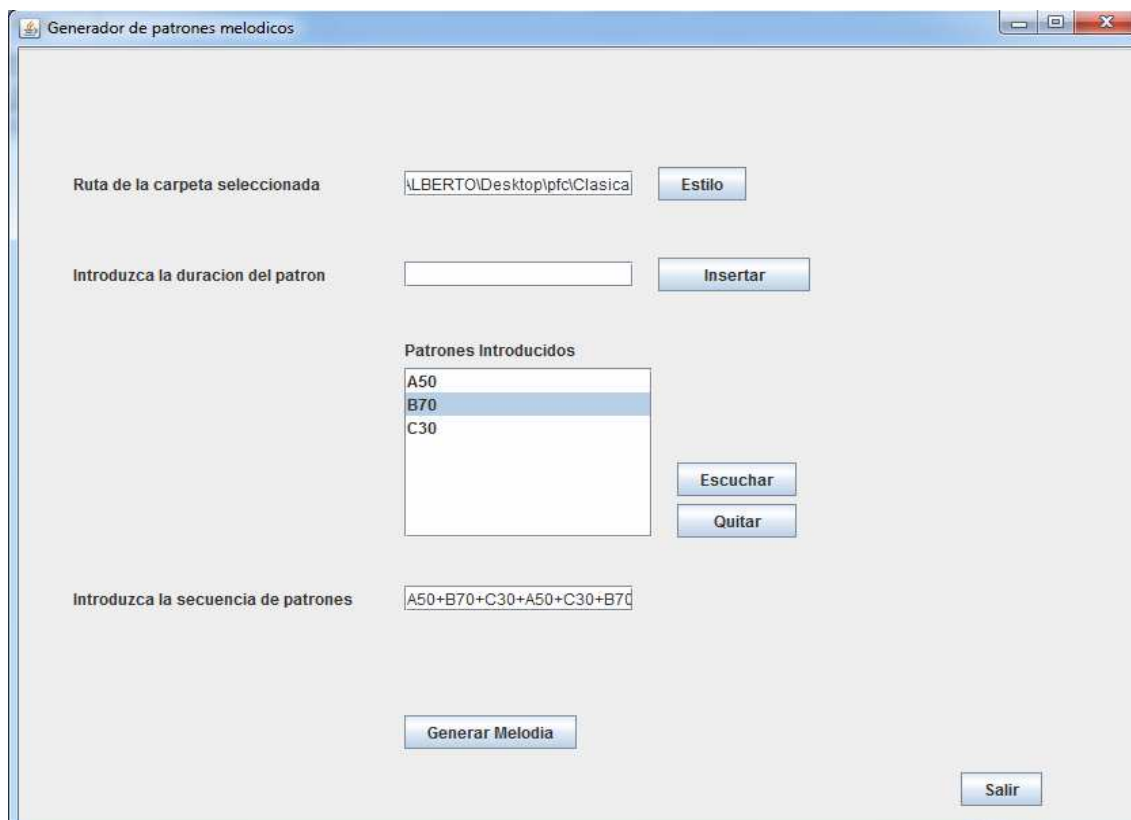


Figura 96 Interfaz con secuencia de patrones elegidos

Como se puede ver, en el campo “Introduzca secuencia de patrones” aparece la secuencia de patrones que se ha elegido. Llegados a este punto, solo queda pulsar el botón Generar Melodía, con lo que se genera el MIDI correspondiente a la secuencia de patrones que hemos elegido.

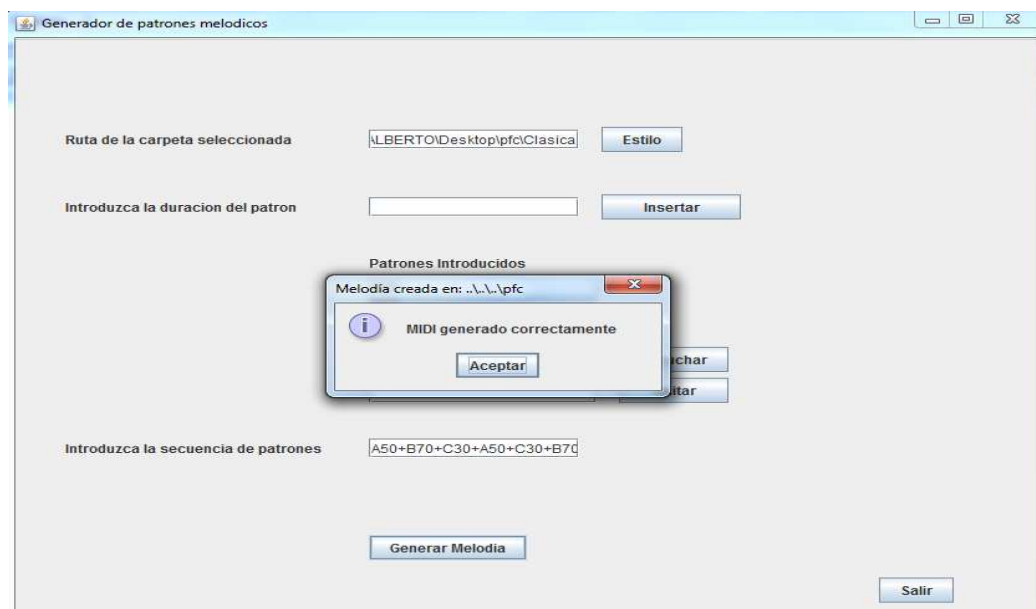


Figura 97 Interfaz con melodía generada

Al pulsar el botón, salta un mensaje que nos avisa de que el proceso se ha ejecutado correctamente e informa al usuario de la ubicación del MIDI resultante, en este caso la carpeta principal de la aplicación.



Nombre	Fecha de modifica...	Tipo	Tamaño
 Clasica_14.05.11.2042	14/05/2011 21:00	Secuencia MIDI	5 KB
 Clasica_14.05.11.2042	14/05/2011 20:42	Archivo TXT	3 KB

Figura 98 Resultado final

Al abrir la carpeta de destino, el usuario encontrará el MIDI generado junto al fichero .txt que sirvió de base para su generación. Cabe decir que los ficheros MIDI generados se distinguen por el estilo más la fecha de su creación.

Apéndice 2: Experimentos no incluidos en los resultados

Experimento 8: Melodía formada por un solo patrón con obras de distintos autores.

-Objetivos

Mediante este experimento se va a observar el resultado de esta mezcla de obras de distintos autores para generar un único patrón, con el fin de observar si aumenta la sensación de contraste al incluir frases de distintos autores en un mismo patrón.

-Preparación

Para la realización de este experimento se han incluido obras de diversos autores, como Mozart, Chopin, Beethoven, Vivaldi, etc. El objetivo es generar un midi compuesto por un solo patrón. Este patrón tendrá una longitud equivalente a 250 negras y estará formado por frases pertenecientes a cualquiera de las obras incluidas.

-Resultados



Figura 99 Melodía obtenida en Experimento 8

Como se puede apreciar en la partitura, el resultado obtenido no incluye cambios bruscos respecto a intervalos de altura o duración. Esto es debido al mismo aspecto que se comento para este mismo experimento realizado con una sola obra, ya que cada frase toma como nota y figura inicial la última de la frase anterior, produciendo un sonido más uniforme.

Sin embargo, si se pueden apreciar cambios respecto a la duración de las figuras y la altura de las notas de un modo progresivo a lo largo de la composición, como se puede ver en los siguientes compases:



Figura 100 Fragmento primero Experimento 8



Figura 101 Fragmento segundo Experimento 8



Figura 102 Fragmento tercero Experimento 8

Viendo estos compases pertenecientes al midi obtenido, se puede ver como han logrado captarse distintas frases dentro del mismo patrón. Para ello ha ayudado el aumento de frases a elegir a la hora de generarse la melodía, ya que al incluir una selección mayor de partituras el número de frases de duración y de altura se incrementa notablemente. Como contrapunto a la inclusión de tantas frases variadas, el resultado obtenido es poco uniforme y hay una sensación elevada de contraste, al incluir frases de obras de distintos autores de modo que su unión resulta poco natural. Esta sensación aumenta al no incluirse en la composición repetición de estructuras, lo que no propicia una sensación de familiaridad al escucharla.

Experimento 9: Melodía formada por patrones de duración media con obras de distintos autores.

-Objetivos

El objetivo de este experimento es conseguir una melodía con una mayor variedad rítmica y musicalidad que la que es posible conseguir con un solo patrón. Para ello se han generado unos patrones con una rítmica distinta entre si para comprobar el resultado que se obtiene al combinarlos. Además, se va a comprobar el efecto de combinar patrones formados por frases de distintas obras para evaluar el resultado obtenido.

-Preparación

Para la realización de este experimento se van a generar varios patrones de duración media para construir una melodía a partir de ellos. En concreto se van a generar 3 patrones de 24 negras de duración (patrones A, B, y D) y 1 patrón más de duración 16 negras (patrón C). La estructura de estos patrones se muestra a continuación:

Patrón A



Figura 103 Patrón A Experimento 9

Patrón B



Figura 104 Patrón B Experimento 9

Patrón C



Figura 105 Patrón C Experimento 9

Patrón D



Figura 106 Patrón D Experimento 9

Se va a seguir la siguiente secuencia de patrones a la hora de crear la melodía:

ABABCABDCABAB

De modo que la estructura principal la formen los patrones A-B y los patrones C y D sirvan como puente para volver a llegar a esta estructura.

-Resultados

The musical score is written for guitar in 4/4 time. It consists of 32 measures, numbered 1 through 32 in red. The notation includes various rhythmic patterns such as eighth notes, sixteenth notes, and quarter notes, along with rests and accidentals. The score is organized into ten staves, with measures 1-4 on the first staff, 5-8 on the second, 9-13 on the third, 14-16 on the fourth, 17-20 on the fifth, 21-25 on the sixth, 26-27 on the seventh, 28-30 on the eighth, and 31-32 on the ninth. The piece concludes with a final double bar line at measure 32.

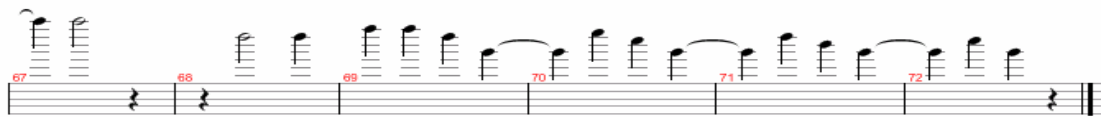


Figura 107 Melodía obtenida Experimento 9

El resultado obtenido se puede ver en la partitura que figura arriba. Al igual que en el experimento de similar estructura realizado anteriormente, se observa que el generar una composición con unas estructuras que se repitan hace que suene más agradable al oído y mucho menos caótico.

Al generar los patrones que formarían parte de la composición, se buscaban patrones que por un lado tuvieran figuras de poca duración y variaciones en la altura de las notas notables, y por otro lado patrones que fueran lo contrario, figuras de duración elevada y poca variación en la altura de las notas. Los patrones A y el C responden al primer criterio y los patrones B y D al segundo. El resultado de esta elección es una melodía con unos cambios rítmicos más fuertes y una sonoridad más variada. Sin embargo, esta unión suena poco natural en ciertos momentos de la composición, como los compases comprendidos entre el 25 y el 36, donde se unen los patrones C y A. El resultado de unir estos dos patrones con figuras de poca duración y mucha variación de notas resulta algo atropellado y poco natural al escucharlo con el correspondiente acompañamiento.

Cabe decir que la obtención de patrones que sonasen diferentes no ha resultado una tarea muy costosa ya que al incluir partituras de muchos autores el número de frases ha aumentado considerablemente, contando con unas 2608 frases para este experimento. Esto ha propiciado que con pocas ejecuciones del programa se obtuviesen patrones distintos entre si, ya que al tener una gran muestra de frases la probabilidad de generar patrones distintos aumenta considerablemente.